UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT D

COPY

Form Approved
OMB No 0704-0188
Exp Date Jun 30 1986

1a REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

AD-A226 193

2a SECURITY CLASSIFICATION AUTHORITY

ITY OF REPORT

2b DECLASSIFICATION / DOWNGRADING SCHEDULE

RELEASED WITH UNLIMITED RIGHTS TO THE GOVERNMENT

4 PERFORMING ORGANIZATION REPORT NUMBER(S)
GF006-F

5 MONITORING ORGANIZATION REPORT NUMBER(S)
ARO 26842.1-EL-SBI

6a NAME OF PERFORMING ORGANIZATION
SIGNAL ANALYTICS CORP.

6b OFFICE SYMBOL (If applicable)

7a NAME OF MONITORING ORGANIZATION
US ARMY RESEARCH OFFICE

6c ADDRESS (City, State, and ZIP Code)
374 MAPLE AVE EAST, SUITE 200
VIENNA, VA 22180-4718

7b ADDRESS (City, State, and ZIP Code)
P. O. BOX 12211
RESEARCH TRIANGLE PARK, NC 27709-2211

8a NAME OF FUNDING / SPONSORING ORGANIZATION

8b OFFICE SYMBOL (If applicable)

9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
DAAL03-89-C-0033

8c ADDRESS (City, State, and ZIP Code)

10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
A METHOD OF ANALYZING ATR SYSTEM PERFORMANCE BASED ON SHAPE DISTORTION

12 PERSONAL AUTHOR(S)
MORT, DR. MICHAEL STEVEN

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
FINAL	FROM 8/24/89 - 4/30/90	90-05-04	56

16 SUPPLEMENTARY NOTATION

17 COSATI CODES		
FIELD	GROUP	SUB-GROUP

18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Thermal performance model, Infrared imaging, Image processing, Automatic target recognition, Modulation transfer function, Detection and recognition perfor

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

SEE NEXT PAGE.

DTIC
ELECTE
AUG 31 1990
S B D
Co

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS

21 ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED

22a NAME OF RESPONSIBLE INDIVIDUAL

22b TELEPHONE (Include Area Code)

22c OFFICE SYMBOL

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Signal Analytics
Innovative solutions in signal and image processing

A METHOD OF ANALYZING ATR SYSTEM PERFORMANCE BASED ON SHAPE DISTORTION

FINAL REPORT

Contract No. DAAL03-89-C-0033

Submitted to

US Army Research Office

May 5, 1990

Contractor:	Signal Analytics Corporation 374 Maple Avenue East, Suite 200 Vienna, VA 22180-4718 (703) 281-3277
Contract Title:	A Method of Analyzing ATR System Performance Based on Shape Distortion
Contract Number:	DAAL03-89-C-0033
CLIN:	0002AB: FINAL Fiscal Report
Period:	February 24 through May 4, 1990
Author:	Dr. Michael Mort
Date of Report:	May 7, 1990

Number of direct labor hours expended on the contract in this period:	148
Travel Charges:	$0.00
Other Direct Charges:	$0.00
FINAL Monthly Payment Due:	$8,993.00
Total Previous Billings:	$44,950.00
Expected funding required to complete contract:	$0.00

Abstract

The goal of this research was to develop a computational method for modeling automatic image processing system performance. Like the Minimum Resolvable Temperature (MRT) measure in common use for man-in-the-loop imaging systems, this new method should have the following characteristics:

(i) It is easy to compute.

(ii) Results based on system performance with a synthesized pattern, but extendable to be independent of the target pattern.

(iii) Gives performance results as a function of the sensor design parameters.

In this research we have developed a measure, which we call the Maximum Resolvable Polygon (MRP), which estimates the degree of shape distortion introduced by an imaging system. The MRP relies on a computer simulation of the imaging system. While such a computer simulation requires more extensive computer power than the computation of an analytical formula like the MRT, the MRP may still be considered relatively easy to compute. All software simulations and experiments reported here were performed on a personal computer.

A new test pattern set has been proposed to replace the bar pattern which is used to analyze the performance of man-in-the-loop systems. Target-like test patterns rely on the use of regular polygons and are parameterized by the number of sides in the polygons. Clutter-like objects are also derived from this test pattern set.

A theoretical analysis of the MRP indicates that it estimates the target shape conditions at which a target recognition system will produce a 50% error rate. Simulation experiments have compared the performance which the MRP measure predicts that an automatic target recognition system would have with the performance of the human visual pattern recognition system. These experiments indicate that the MRP can be used as a reliable measure of the relative performance ability of various imaging systems. Although performance results are not derived analytically, the simulation analyses do produce empirical relationships between performance and sensor system design parameters.

The primary application of this research will be to assist in the design and development of new sensor systems. The basic performance modeling technique developed here can be used to perform sensor system trade-off studies on resolution versus sensitivity, detector shape, and sampling array geometry. Extensions may be made to other system design parameters.

Signal Analytics

Innovative solutions in signal and image processing

A METHOD OF ANALYZING ATR SYSTEM PERFORMANCE BASED ON SHAPE DISTORTION

FINAL REPORT

Contract No. DAAL03-89-C-0033

Submitted to

US Army Research Office

May 5, 1990

Table of Contents

1. Abstract

The goal of this research was to develop a computational method for modeling automatic image processing system performance. Like the Minimum Resolvable Temperature (MRT) measure in common use for man-in-the-loop imaging systems, this new method should have the following characteristics:

(i) It is easy to compute.

(ii) Results based on system performance with a synthesized pattern, but extendable to be independent of the target pattern.

(iii) Gives performance results as a function of the sensor design parameters.

In this research we have developed a measure, which we call the Maximum Resolvable Polygon (MRP), which estimates the degree of shape distortion introduced by an imaging system. The MRP relies on a computer simulation of the imaging system. While such a computer simulation requires more extensive computer power than the computation of an analytical formula like the MRT, the MRP may still be considered relatively easy to compute. All software simulations and experiments reported here were performed on a personal computer.

A new test pattern set has been proposed to replace the bar pattern which is used to analyze the performance of man-in-the-loop systems. Target-like test patterns rely on the use of regular polygons and are parameterized by the number of sides in the polygons. Clutter-like objects are also derived from this test pattern set.

A theoretical analysis of the MRP indicates that it estimates the target shape conditions at which a target recognition system will produce a 50% error rate. Simulation experiments have compared the performance which the MRP measure predicts that an automatic target recognition system would have with the performance of the human visual pattern recognition system. These experiments indicate that the MRP can be used as a reliable measure of the relative performance ability of various imaging systems. Although performance results are not derived analytically, the simulation analyses do produce empirical relationships between performance and sensor system design parameters.

The primary application of this research will be to assist in the design and development of new sensor systems. The basic performance modeling technique developed here can be used to perform sensor system trade-off studies on resolution versus sensitivity, detector shape, and sampling array geometry. Extensions may be made to other system design parameters.

2. Introduction

Imaging sensor systems have been used primarily in man-in-the-loop systems where a human operator must view the image created by the sensor and make decisions based upon the image. As a result, sensor system parameters have been optimized for human visual detection and recognition capability. However, as Automatic Target Recognition (ATR) becomes more widespread, sensor design specifications must take into consideration the requirements of the ATR algorithms to achieve optimal performance. In order to perform engineering trade-offs and evaluation of competitive sensor designs, models are required which predict algorithm performance as a function of the sensor system parameters. Such performance models also allow engineers to determine the limiting performance of a sensor system due to phenomenological variations. For example, even if a detector of infinite resolution and sensitivity could be devised, atmospheric turbulence and temperature fluctuations will limit the ability of the system to recognize distant targets. Knowledge of such limiting performance allows the system designer to place practical bounds on the sensor parameters and avoid needless over-specification (and higher cost).

Although the affects of sensor parameters on human recognition performance have been studied extensively, work done to relate sensor parameters to ATR performance is still in its infancy. For example, a 30 Hz field rate of interlaced video was settled upon for man-in-the-loop systems because the integration time associated with human vision dictates that this is nearly the minimum frame rate to avoid annoying flicker. However, an ATR system would not be bothered by flicker. There are other considerations which would determine the optimum frame rate for ATR systems, including the trade-offs between blurring due to motion in the scene or of the camera and noise reduction due to longer detector integration time.

System performance analysis is the study of how information is degraded through a system. The classical system performance analysis criterion for man-in-the-loop image processing systems has been the Night Vision Laboratory Static Performance Model.[1] This model utilizes an analytical formulation of information degradation based on effects of the system on the frequency components in the image data. Since it is a frequency domain effects model, it is primarily useful for analysis of the linear-shift invariant components of an imaging system, such as the optics and electronics at the front end. The model gives performance results in terms of the Minimum Resolvable Temperature Difference (MRT). The MRT has the following significant features:

(i) It is easy to compute.

(ii) Its results are based on system performance with a synthesized pattern (a bar pattern), but they are extended through empirical analysis to be independent of the target pattern.

(iii) It gives performance results as a function of the sensor design parameters.

In man-in-the-loop systems, all information processing is performed by the human, who is modeled as a linear filter matched to a signal of the desired frequency. Modern imaging systems contain several more components which may not be modeled as linear shift invariant components. For example, staring focal plane arrays have two dimensional sampling inherent in the architecture of the system (Sampling is linear but not shift invariant, hence it cannot be represented as a convolution or as multiplication in the frequency domain.). Also, in the case of an ATR image

---

[1] J. Ratches, et al, "Night Vision Static Performance Model for Thermal Viewing Systems, US Army Night Vision Laboratory Technical Report Number ECOM-7043, April 1975.

processing system there is another major subsystem components to consider, the algorithm subsystem, which may be highly nonlinear. For this reason, alternatives to frequency-based performance analyses are being considered.

The principal impediment to progress in the development of models of ATR system performance is the lack of a measure of ATR system performance having the same characteristics as MRT. The problem is that any imaging system has several components which contribute to the degradation of the information† contained in the image data. This information degradation introduced at various stages in the system are naturally represented in different forms: pixel intensity variations (optical MTF and noise), shape variations (segmentation errors), variations in equivalence classes of the computed features (insufficient feature set), and decision error probability (system output). In order to have a systematic performance model, these various forms of information degradation must be related to a single measure.

Historically, researchers have used the natural representation of information degradation at the output of the ATR system, namely decision error probability, as the measure of effectiveness of the entire ATR system. The difficulty with this approach is that there are a wide variety of ATR algorithms which are developed more or less heuristically, involving a sequence of many steps. The nonlinearity and procedural nature of many algorithms make a rigorous end-to-end probability analysis intractable. Thus there are basically two ways to proceed with system performance analysis if decision error probability is chosen as the measure: Monte Carlo simulations, or information theoretic methods (i.e. mathematically defined information) with approximations.

End-to-end Monte Carlo simulation appears to be the prevailing technique used to evaluate image processing system performance . There are several shortcomings with this approach. It is difficult to extrapolate the results on a fixed target set to other target types. The entire simulation must be run again if a new sensor is used since there is no way of relating the simulation results directly to the parameters of the sensor system. Finally, end-to-end simulations do not provide insight into the characteristics of the image data or the sensor which led to the success or failure of the test.

There has also been some research into the use of mathematical-information theoretic methods to find theoretical bounds on decision error probability. These methods seek to avoid performing Monte Carlo simulations on a large number of images by performing complicated computations on the mathematical-information content of the image data rather than on the image data itself. Such methods rely on complicated transformations of the image data to determine the mathematical-information content of an image. Furthermore, since algorithms are defined to operate on the image data, this method requires that the algorithms be re-described in terms of their effects on the mathematical-information in the image. This step requires such extensive simplifying assumptions that one wonders whether one is actually modeling the performance of any real algorithm. Thus the use of mathematical-information theoretic methods tends quickly to become mathematically intractable and non-intuitive.

In the research reported on here, we have taken the view of image processing system performance modeling that accurate models of the imaging system and actual algorithms should be used whenever possible. Usually this means using computer software descriptions of both the sensor and the algorithms. It is undesirable to describe the effect of some processing step on the mathematical-information theoretic content of an image when the algorithm is actually applied to the image data. However, it is equally undesirable to have to perform an end-to-end Monte Carlo simulation each time a parameter in the sensor is changed. In order to address this dilemma,

---

† Unless otherwise indicated, we use the term "information" in the ordinary sense of human communication, not in the mathematical sense of Shannon's Information Theory.

innovative performance modeling techniques must be considered. Thus we have investigated an approach which is neither a completely abstract analytical model of the system performance nor a full Monte Carlo simulation.

While previous attempts to define a performance model have concentrated on relating all the system distortions to the measure of distortion which is used at the output of the entire system, i.e. decision error probability, in this research we have investigated the possibility of relating all these distortions types to one of the other natural representations: Object Shape. Choosing shape distortion as a measure of performance would force imaging systems to concentrate on retaining shape fidelity for Automatic Target Recognition systems in the same way that current imaging systems are forced to image bar patterns accurately in order to obtain a good MRT score.

The objective of this research was as follows:

Objective: Develop a computational method for modeling automatic image processing system performance. Like the MRT measure for man-in-the-loop systems, this method should have the following characteristics:

(i) Easy to compute.

(ii) Results based on system performance with a synthesized pattern, but extendable to be independent of the target pattern.

(iii) Gives performance results as a function of the sensor design parameters.

In addressing this objective, we have concentrated on image processing systems which use a single frame of image data from a single monochromatic sensor to Detect, Classify, Recognize, or Identify a single target within the sensor field of view.

The results of this research are summarized below:

i) We developed an imaging system performance analysis method based on measuring how each component of an image processing system distorts the shape of a target or clutter object.

ii) We related shape distortion to probability of error.

iii) We developed computer software to implement the computation of the shape distortion effects in an imaging system. This computer software uses existing computer models for defining system components, whenever they are available, with as little modification as possible.

iv) We developed a parametric set of test patterns which is similar to real target and clutter patterns so that performance results may be achieved by analyzing only a small set of data.

v) We used our new approach in two comparisons of imaging sensors: (i) rectangular arrays vs hexagonal arrays of detector elements, and (ii) varying sensor signal to noise ratios.

3. Shape Distortion as a Measure of Imaging System Performance

3.1. Analyzing Imaging System Performance

As mentioned above, system performance analysis is the study of how information is degraded through a system. In an ATR system there are two major subsystem components to consider: The sensor subsystem and the algorithm subsystem. In order to obtain a consistent measure of imaging system performance, the inter-related effects of information degradation introduced by all components of the system must be understood and considered. There are two principal sources of image degradation in the imaging sensor itself: resolution degradation, which is usually modeled as deterministic and signal dependent, and additive noise which is usually modeled as random and signal independent (at least to a reasonable approximation). After the image is formed, the algorithm subsystem extracts information from the image. These algorithms can generally be decomposed into four components: preprocessing, segmentation, transform (feature) computation, and feature matching (i.e. pattern recognition).

Each component of an imaging system generally has a computer model or algorithm description which depends on certain design parameters. In fact in many cases, a system component may be highly nonlinear or may be described only in terms of mathematical or computer software descriptions (e.g. think of some complex scheme of thinning the edges found after edge detection). This presents severe problems for a totally analytical approach to system performance analysis.

In light of these comments, we have decided to study how information is degraded through a system which is described in terms of such models and codes with as little modification as possible to the mathematical-algorithmic descriptions of the system. Therefore, we have chosen to utilize a computer simulation of the imaging system components. Figure 1 shows the nature of the simulation. The upper path is the simulation of the system components. Each of the simulated components operates on a high resolution digital representation of the test pattern. The output of each system component is passed to the next simulated system component as well as to a distortion computation block. By comparing the distortions after each system component, the information loss through the system can be monitored.

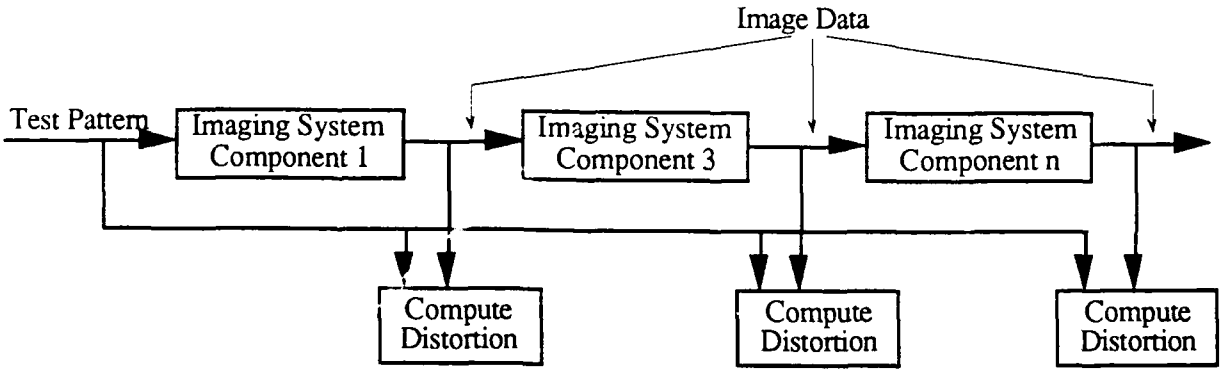


Figure 1. Block Diagram of the System Simulation and Shape Distortion Computation

The information degradation introduced at various stages in the system are naturally represented in different forms: pixel intensity variations, shape variations, variations in equivalence classes of the computed features, and decision error. In order to have a systematic performance model, these various forms of information degradation must be related to a single measure. Previous attempts to define system performance models have concentrated on relating all the distortions to the measure

of distortion which is used at the output of the entire system, i.e. decision error. The object of this research has been to investigate the possibility of relating all these distortions types to one of the other natural representations: Object Shape. Several issues must be addressed in determining how to use the object shape distortion as a measure of imaging system performance:

i) We must determine a set of test patterns which is similar to real target and clutter patterns so that performance results may be achieved by analyzing only a small set of data.

ii) We must adopt a shape representation that:

a) Allows us to determine how each of the system components affects object shape, given some accurate model of the imaging system components in the form of mathematical or computer software descriptions; and

b) Permits an efficient method of computing the distortion in the object shape at the output of each of the system components.

iii) We must relate object shape distortion to the probability that the imaging system makes an error.

These issues are addressed in the subsequent sections.

3.2. The Test Pattern Set and Definition of the Imaging System Tasks

One of the goals of the research was to obtain performance results which are based on the use of synthesized patterns but which are extendable to be independent of the target pattern. Synthetic imagery is most useful for experimentation since it allows more accurate control of the computations than is possible with real targets. The NVL Static Performance Model uses a pattern of bars, since it relates information content of an image to its frequency content. Since we are exploring a different approach which relies on shape distortion, we require a set of test patterns with more shape definition. In the tactical use of Army ATR systems there are a large variety of targets and clutter. What is sought for performance modeling is a set of patterns with characteristics which make them similar, in some sense, to real targets and real clutter, but which are easier to manipulate than real target and clutter signatures. We have developed a parametrically defined pattern set which we believe can be used as the basis for target and clutter independent performance analysis.

3.2.1. Target Shapes

An observation about targets motivates our choice of target shapes. Targets generally are man made objects which tend to have regular shapes (in the sense of being orderly, or neat). The most regular of all shapes is the Circle, which we use as the baseline target shape. Many tactical targets can be geometrically described as the union of a collection of polygons. Therefore, additional target shapes are generated by regular polygons: isosceles triangle, square, pentagon, etc.

Each of these shapes is convex, and their boundaries can be defined in polar coordinates by a radius function $\rho(\theta)$. We use a polar coordinate definition of these shapes because it provides a compact way to define them parametrically, and because rotated versions of the shapes can be generated very easily.

3.2.1.1. Circle

The formula for the boundary of the circular target is given by

$$\rho(\theta,\infty) = r,$$

r is the constant radius of the circle.

3.2.1.2. Regular Polygon

The formula for the regular polygon with n sides in polar coordinates is given as follows (see the Appendix for the derivation of this formula):

$$\rho(\theta,n) = r_n/\cos(\theta - 2\pi/2n - k\,2\pi/n), \text{ for } k\,2\pi/n \le \theta \le (k+1)\,2\pi/n, \; k=0,1,....n-1$$

where r_n is the minimum radius of the n-sided regular polygon, n is the number of sides in the polygon, and k is an index for which side is being drawn, 0 corresponding to the first side, n-1 corresponding to the last side. This formula can be simplified slightly by using modulo arithmetic:

$$\rho(\theta,n) = r_n/\cos(\,(\theta \bmod (2\pi/n)\,) - 2\pi/2n\,)$$

The expression x mod y for x and y being real numbers is construed to mean the same thing as for integer values of x and y, namely

$$x \bmod y = x - \text{Greatest Integer}[x/y].$$

3.2.2. Clutter Shapes

Clutter shapes tend to be highly varied. With open terrain in the background, one would expect clutter objects to be fairly irregular in shape, with many sharp corners. Some geographical features, however, may look quite smooth, such as the boundary of a water body or road or the sides of buildings. These latter clutter shapes have some characteristics similar to target shapes.

In light of this discussion we derive clutter shapes from the polygonal shapes by randomly modifying the boundary of the polygons as follows:

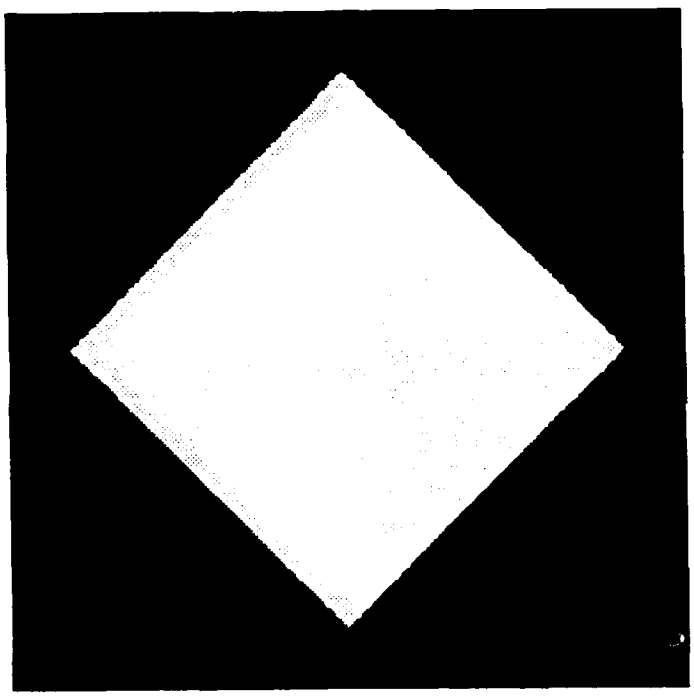
$$\rho_{clutter}(\theta,n) = \rho(\theta,n) + U(\theta,a,b)$$

where U is a random function with the two parameters: a = the magnitude of the random function, and b which relates to the bandwidth of the random function. A low bandwidth random function tends to create more regular looking clutter than is created by using high bandwidth random functions. The random function is added to the target shapes in order to obtain clutter shapes which are approximately the same size as the targets. The random function U is generated according to the following formulation

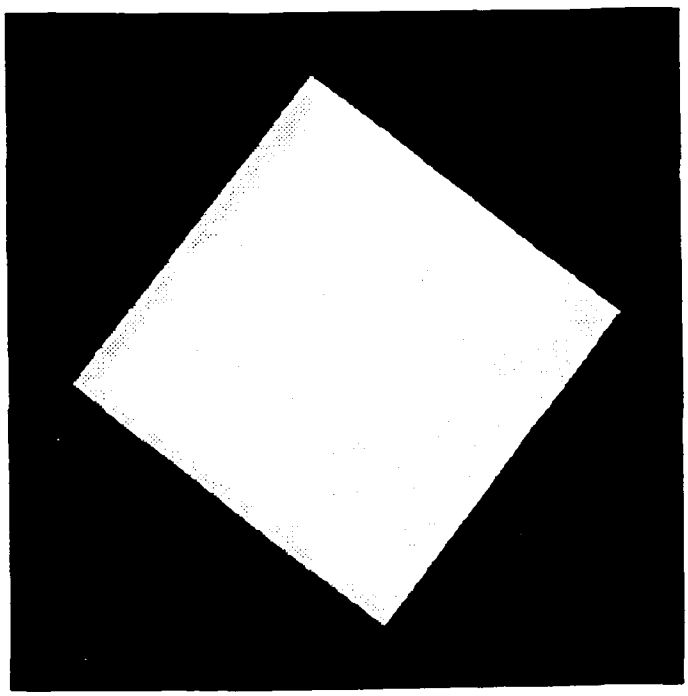
$$U(\theta_k,a,b) = (1-b)*U(\theta_{k-1},a,b) + b*\text{ranu}(k)$$

where ranu(k) is a random number generator giving independent, uniformly distributed random values in the range [-.5,.5], and θ_k are samples of the angle parameter θ. The closer b is to unity, the wider the bandwidth of the resulting function U becomes.

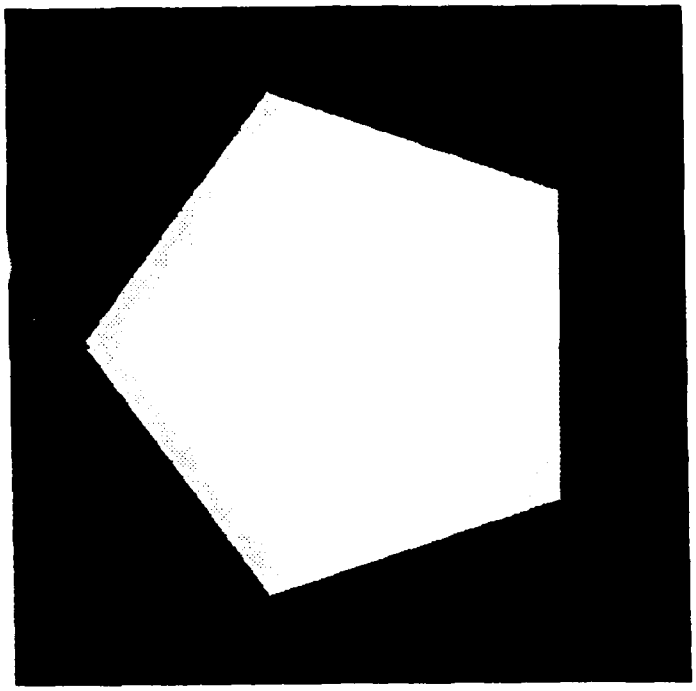
Figure 2 shows images of several polygonal targets and clutter shapes. Notice that it is easy to generate rotated versions of the targets and clutter shapes by adding a constant rotation angle to θ in the above formulas. The ability to create objects with varying rotations is useful because some imaging systems may have "preferred orientations" in which they give better results than in other orientations.



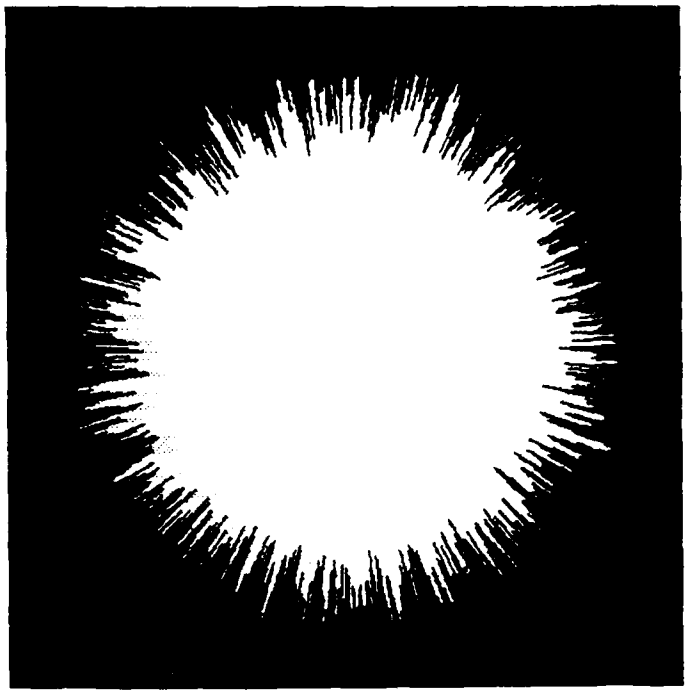
a)



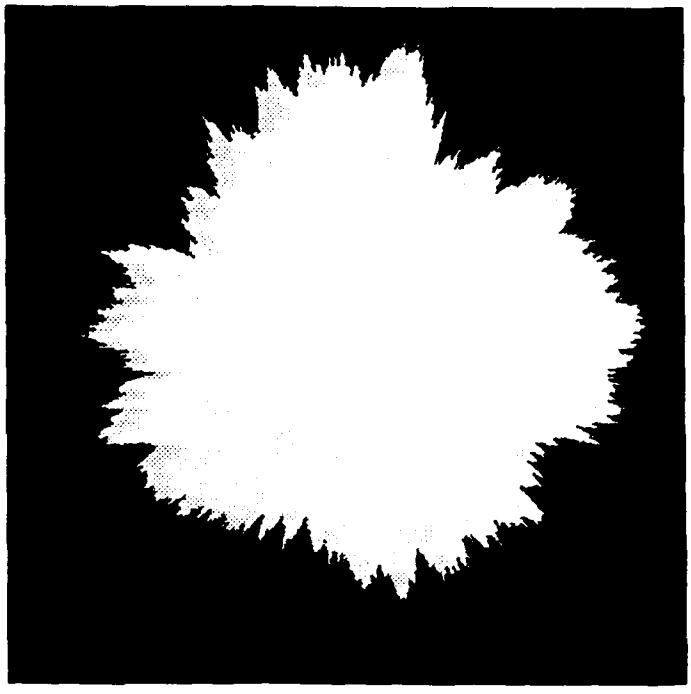
b)



c)



d)



e)

Figure 2. Polygonal Target and Clutter Shapes: a) 4-sided polygon, b)Rotated 4-sided polygon, c) 5-sided polygon, d) High bandwidth clutter object created with $a = 50$, $b = .8$, e) Low bandwidth clutter object created with $a = 400$, $b = .02$.

3.2.3. Definition of Detection, Recognition, and Identification

We define the tasks of the imaging system within the context of these target and clutter shapes to be as follows

i) Detection: Distinguish between the polygonal shapes and one of the clutter shapes derived from the circle.

ii) Recognition: Distinguish between one of the polygonal shapes and the circle shape.

iii) Identification: Distinguish between two different polygonal shapes, or equivalently, count the number of sides in a polygonal shape.

The single parameter, the number of sides in a polygonal shape, characterizes the difficulty of the target set for recognition or identification. For example, increasing the number of sides in the polygonal target increases the difficulty of the system to distinguish the polygon from the circle. The clutter parameters a and b define the difficulty of the Detection task. The closer the amplitude a and bandwidth b are to zero, the more the clutter object will look like the circle target.

3.3. Shape Representation and Shape Distortion Computation

The original shapes are defined in terms of polar coordinates for convenience in creating them for the simulation. This may not be the most ideal coordinate system to use during the analysis of the system effects on the shapes. There are several desirable characteristics to be considered in choosing an object shape representation scheme:

i) The representation scheme must be able to unambiguously define the desired patterns.

ii) The shape representation must be compatible with the computer software simulation of the system components.

iii) In general, the accuracy of imaging system simulations increases as the resolution of the digital test patterns increases. Since high resolution digital test patterns usually require large amounts of memory, the shape representation scheme should be memory efficient.

iv) The desired shape distortion measures must be computable from the shape representation.

We now discuss two representation schemes with respect to the characteristics stated above. Then we present our reasons for choosing the representation we did.

3.3.1. Shape Representation in Rectangular Coordinates

The most obvious and straightforward method of representing image shapes is simply to provide a digital mask of the test patterns themselves in standard two-dimensional rectangular coordinates. Such an image presentation has the form

$$f(i,j) = \begin{cases} 1 & \text{if } (i,j) \in \text{target} \\ 0 & \text{else} \end{cases}$$

where i and j are the indices in a rectangular lattice. The images of Figure 2 are examples of this type of representation.

Although the polygonal test patterns are each binary valued images which represent the shapes of the test pattern, the output of any component in the imaging system may in fact take on other values besides 0 and 1. Furthermore, shape distortion analysis concentrates on how the imaging system distorts the boundaries of the targets. The values of pixels in the interior of the target are not relevant to shape distortion analysis. Therefore whenever the shape distortion effects of a particular component of the imaging system are to be analyzed, the image data is first segmented, i.e. the target pixels are labeled with value 1 and the background pixels are labeled with value 0. Virtually all ATR algorithms contain some form of segmentation step. Ideally, the segmentation algorithm which is used in the shape distortion analysis would be the same as the one which is used by the ATR system itself. However, it is also valuable to be able to analyze an imaging system in a way which is more or less independent of the details of specific ATR algorithms. Thus some baseline segmentation algorithm must be used by the Shape Distortion Analyzer. The specific segmentation algorithm used in our computation of shape distortion is discussed later.

The rectangular coordinates shape representation scheme clearly enjoys the characteristics (i) and (ii) of a desirable shape representation scheme. Since there is no coding of the image data, this representation of the test patterns is compatible with the standard existing computer software simulations of the system components, and it can obviously be used to represent any desired shape.

This representation scheme also submits to easy computation of at least one useful distortion measure:

$$A_{rect}(n,I) = \int_A |s_n - s_n(I)|$$

where $s_n(I)$ is the segmented result of passing an n-sided regular polygon image through the imager I. It is shown in the next section that this distortion measure can be related to error probability to get an overall system performance measure.

The big disadvantage of this representation of shape is that it is extremely memory inefficient. The memory requirements for representing a target shape in an image of dimensions N×N pixels is proportional to N^2.

3.3.2. Shape Representation in Polar Coordinates

A complete and accurate description of the shape of an object can also be given in terms of the location of each border point, which is the second method we considered for representing object shapes. The edge location may be coded in several ways. For example the x-y coordinates of each border point may be given in rectangular coordinates or the position of just one point on the border may be given together with the chain code to each subsequent point on the border. A polar coordinate representation seems particularly useful. Consider for example a high resolution image of a circular target. The shape of the target can be compactly represented by the radius function

$$f(i,j) = \begin{cases} 1 & \text{if } (i^2 + j^2)^{1/2} \le \rho(\theta) \\ 0 & \text{else} \end{cases}$$

where $\rho(\theta)$ = radius of the target at angle θ, which is the distance from the center of the target to the edge as a function of angle (where "center" is suitably defined, e.g. as the centroid). Thus the shape is completely described by the polar coordinates function $\rho(\theta)$. This description of the

circular target results in a one dimensional waveform of constant value. Figure 3 shows the polar coordinate representation of the circular shape pattern and some polygonal shape patterns.

This representation scheme enjoys characteristics (iii) and (iv) stated above for a desirable shape representation scheme. In fact the primary advantage to this polar coordinate coding of the shape edges is that it is very memory efficient, since it ignores the interior points of an object shape and concentrates on points near the edges. The memory requirements for representing a target shape in an image of dimensions N×N pixels is proportional only to N (in fact the number of edge samples required is approximately $\frac{2\pi N}{(\text{pixels/edge sample})}$). Note also from Figure 3 that the polar coordinates representation for the test targets is periodic. Thus in fact the effects of the imaging system on only one period of the radius function would need to be considered in order to get results over the entire shape.

This representation scheme would also permit a potentially rich collection of shape distortion measures to be considered. A radius distortion function can be computed as the absolute value of the difference between the radius function of the original and the distorted waveforms.

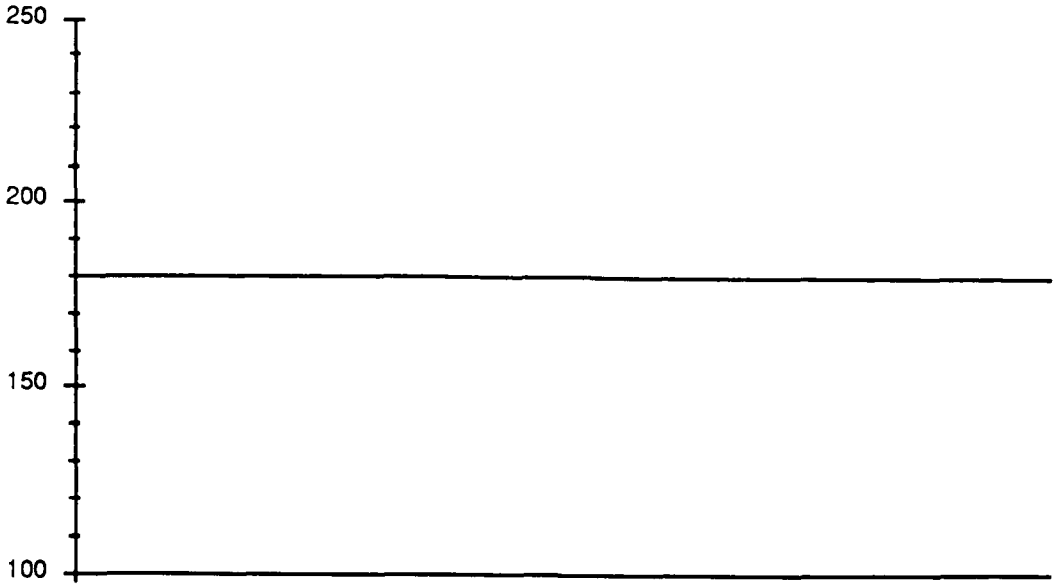
$$d(\theta,n,I) = |\rho(\theta,n) - \rho(\theta,n,I)|$$

where $\rho(\theta,n,I)$ denotes the radius function of the n-sided polygon after distortion by the system I. One obvious scalar measure of distortion is obtained by simply integrating d with respect to θ:

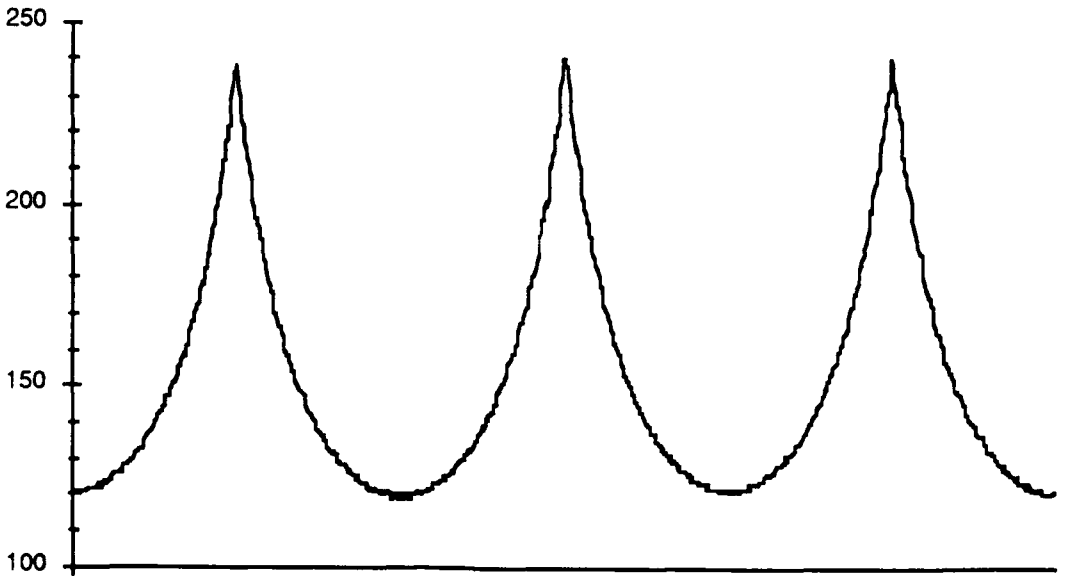
$$A_{polar}(n,I) = \int_{\theta} d(\theta,n,I)$$

This measure is in fact proportional to $A_{rect}(n,I)$ as given above. Other simple and potentially useful summary distortion parameters are also available from $d(\theta,n,I)$. For example the distortion function $d(\theta,n,I)$ may be treated as if it were a random function of θ and statistics may be collected on it, such as the histogram of the values of $d(\theta,n,I)$, or the maximum, mean and standard deviation of its values, or its roughness. Alternatively one may compute the magnitude of the Fourier transform of $d(\theta,n,I)$ (as a function of θ) and use the low frequency portion of the power spectrum (since it is obvious that large values in the low frequency indicate large distortions). While it is not clear that any of these measures are any better at predicting the performance of the imaging system than A_{rect} would be, at least this representation scheme does permit them to be computed very easily from $d(\theta,n,I)$.

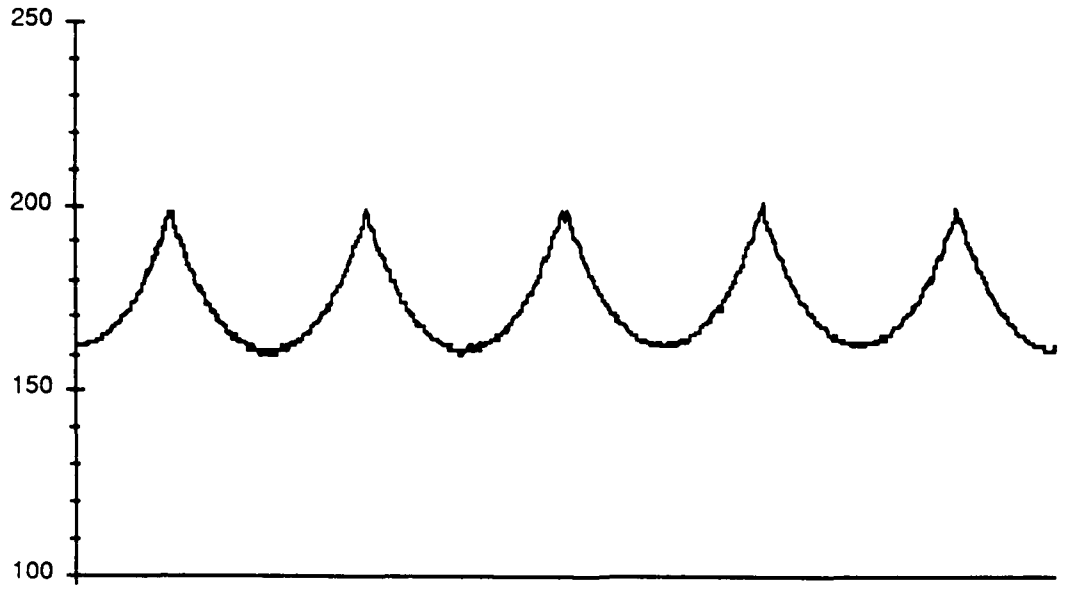
Unfortunately, there are two severe disadvantages to the use of this polar coordinates shape representation scheme. First, it can only be used to represent convex shapes, since the radius function $\rho(\theta)$ is unambiguously defined only for convex shapes. This to not too much of a limitation for the original target and clutter shapes, since each of them is a convex shape which has been defined through such a polar coordinates mapping. However, when sampling and noise are considered, non-convex shapes can result, as will be shown later.



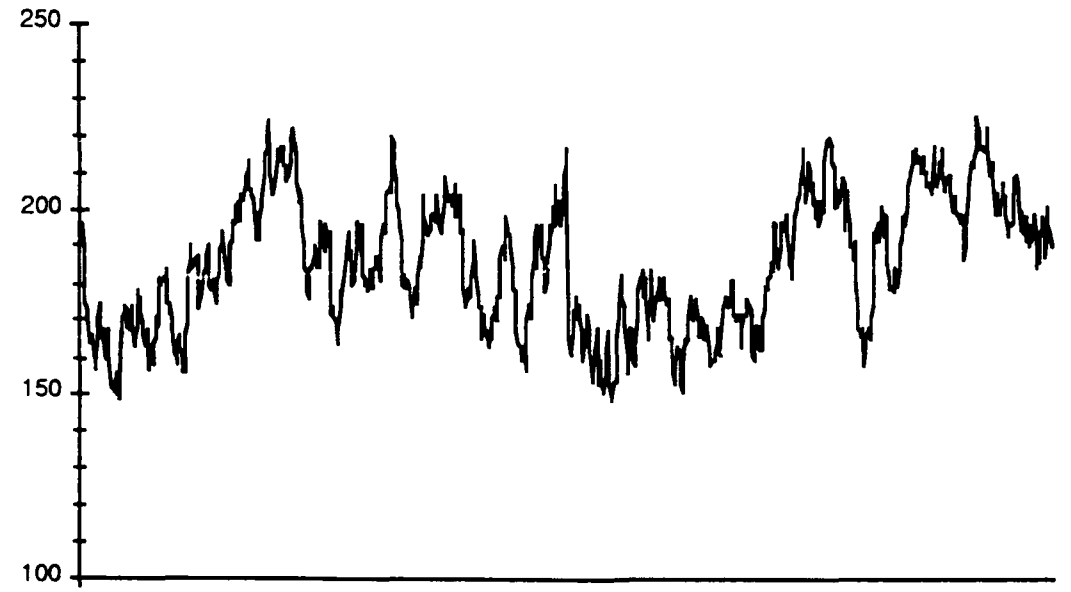
a) Circle



b) 3-Sided Polygon



c) 5-Sided Polygon



d) Low Bandwidth Clutter Object

Figure 3. Polar Coordinate Representations of Selected Shapes.

The most serious serious disadvantage to the use of the polar coordinate representation is the fact that it is not directly compatible with standard existing computer software simulations of the system components. This means that existing mathematical and algorithmic descriptions of imaging system components would have to be modified to accommodate the system simulation shown in Figure 1. Most models for system components assume a rectangular coordinate system representation for the input image data. Thus if we were to replace the standard representation of a circular image with the polar representation given by the simple radius function $\rho(\theta) = \text{const.}$, we would have to determine out how to simulate the optics, the sampler and all the other components of the imaging system in terms of processing this one-dimensional radius function. This is fairly difficult for the filtering and sampling components of an imaging system system, but it would be even more difficult for some exotic algorithmic components of the system.

3.3.3. The Shape Representation Choice

Although it was our original intention to pursue the use of the polar coordinate representation of shapes in this research, we have settleed on using the rectangular coordinates shape representation as given above for the following reasons:

i) The rectangular coordinates shape representation is the most compatible with existing software models of the imaging system. By using the rectangular coordinates shape representation we were able to spend more time on analyzing the effectiveness of using shape distortion as a performance measure and less time on modifying existing models of imaging systems.

ii) The rectangular coordinates shape representation is more readily understandable when it is displayed.

iii) Although rectangular coordinates shape representation requires more computer memory to store and process high resolution test patterns than the polar coordinates representation, this characteristic seems to be the least important of the four criteria of a good shape representation scheme which were listed at the beginning of section 3.3.

iv) As we show below, A_{rect} can be directly related to probability of recognition error, which would be substantially more difficult for any of the exotic distortion measures mentioned above which may easily be computed from $d(\theta,n,I)$. Furthermore, there is no clear evidence that any of those other distortion measures are any better at predicting the performance of the imaging system than A_{rect} would be.

Now that we have settled on a shape representation scheme, we show in the next section how the shape distortion computation can be related to imaging system performance in terms of error probability.

3.4. Shape Distortion Computation and Its Relationship to System Error Probability

While probability of error is usually considered the most appropriate measure of system performance for the tasks defined in the previous section, as we have stated, probability of error is often difficult to compute. What is required is a summary distortion measure which is a scalar quantity, which can be functionally related to the probability that an ATR system will successfully perform one of the three tasks detection, recognition, or identification. In this respect the summary distortion measure is to be like the MRT (Minimum Resolvable Temperature) which characterizes man-in-the-loop IR imaging systems.

A bound on system error probability is actually very easy to obtain once we are given the target set. This bound is related to the concept of the Matched Filter. When the target shape is known exactly, the Matched Filter is the best possible linear pattern matching technique. Thus the error incurred by using Matched Filters to perform pattern recognition can be used to provide a lower bound on the error which would be incurred by use of any linear pattern matching technique. The Matched Filter only provides a bound on system performance because in most ATR applications, the exact target shape is not known beforehand. In fact the reason for most of the feature transformations found in ATR schemes (chain code, invariant moments, etc.) is that there are target shape uncertainties due to such unknown transformations as rotation and size scaling of the target. By providing a link between shape distortion analysis and the Matched Filter, we expect to obtain a bound on performance which applies to most statistical pattern matching techniques that use feature transformations of the image.

We stressed the word "linear" above because many features and pattern recognition techniques now under consideration by artificial intelligence researchers are far from being linear. For example, when trying to distinguish between the typewritten letters "t" and "f," one would most likely look for the presence of a rightward curvature at the top or the bottom of the character instead of performing a Matched Filter over the entire character. Similarly one might look for wheels or a gun to distinguish between a truck and a tank. While a bound on system performance based on the linear Matched Filter may appear to be useless to these "AI" techniques, we note that the process of looking for portions of a target is just a collection of subproblems of looking for a lot of smaller targets. Therefore we would apply the concept of the Matched Filter bound to each of these subproblems. How these performance bounds on each subproblem accumulate to form a performance bound on the overall target recognition task depends on how the ATR algorithm combines the results from each of the subproblems. It is unlikely that any generally applicable technique exists to find the overall bound in every such algorithm specific example and we have not investigated this aspect of the problem in this phase of the research.

3.4.1. Background

In target detection, "error" usually means the sum of two types of errors: Miss, and False Alarm. This assumes that there is one class of objects called "Target" and everything else is "Not Target," i.e. "Clutter." In target recognition there are two are more specific classes of objects, say Target A and Target B, and samples must be assigned to one of them. In this case "error" means the sum of all occurrences where either an A sample is assigned to class B, or a B sample is assigned to class A.

Decisions in either Detection or Recognition are often reduced to the problem of thresholding a single functional value which is computed from the image data. From classical detection and estimation theory, this functional is usually given by the output of a Matched Filter. If there is any randomness in the received data, then the functional value is also random. The thresholding value(s) can be set for either the Detection or Recognition task by considering the probability density function (pdf) of the functional value under the various conditions.

The pdf of the Matched Filter output for the target often has some spread around a central value. Similarly, the pdf for the "Clutter" class has some spread. Although some output values may be more or less likely than others in reality, the class "Clutter" is so broad and ill-defined, that one seldom knows anything about its probability density function. This demonstrates a fundamental difference between "Target" and "Clutter." The pdf spread for Target is due entirely to the randomness in the receiver, not due to any uncertainty in the definition of the class. That is, in the absence of receiver randomness (noise), the Matched Filter output for the Target would be a constant. In fact the pdf of the Matched Filter output when noise is present is simply the pdf of the receiver noise, shifted so that its mean corresponds to this value of what the Matched Filter output would be without the receiver noise. The spread of the pdf for Clutter is due mostly to uncertainty

in the definition of the class. Even if there were no noise in the receiver, there would still be a spread (and probably a broad one) for the Clutter pdf. In either task, Detection or Recognition, the definition of the error probability uses the pdf's of receiver noise. Furthermore, in the Detection task, a pdf must be defined for the Clutter class.

The mathematics for computing error probability for recognition of targets in additive white Gaussian noise is well worked out.[2] We summarize those results. Suppose the objective is to decide between two targets, s_a and s_b, one of which is present in an image with additive white Gaussian noise. Define

$$E_a = \int_A s_a^2$$

$$\rho = \int_A s_a \, s_b$$

σ^2 = variance of the additive noise

$$d^2 = \frac{1}{\sigma^2} \int_A (s_a - s_b)^2$$

$$\text{erfc}^*(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$$

Then the error probability is given by

$$P_{error} = \text{erfc}^*(d)$$

It is important to notice that the computation for probability of error depends on the signals involved.

3.4.2. The Maximum Resolvable Polygon for Noise-Free Imaging

We start with the quantity d^2 and use it to compute a summary measure of system performance based on the polygonal and circular target set.

It makes sense to separate the distortion effects of diffraction and sampling from the effects of noise in any imaging system. The deterministic distortion effects should be easier to handle analytically and for some applications, they may be the limiting effects. Thus we defer the discussion of how to handle noisy imaging systems.

---

[2] H.L. van Trees, *Detection, Estimation, and Modulation Theory, Part 1*, John Wiley and Sons, 1968, pp 254ff.

Consider the Recognition problem, i.e. the situation where the targets to be distinguished are fixed and deterministically defined. Since there is no randomness in the receiver and the signals are deterministic, a natural question then is: What does "probability of error" mean where no randomness exists? Considering the above discussion on probability of error, the quantity d would be infinite in a noise free situation since the noise variance σ^2 would be considered 0 in such a case. Inserting this value for d in the formula for P_{error} then gives 0. This is clearly not helpful. Thus we take a different view of the situation. For a receiver in which the noise variance σ^2 is fixed, the quantity of interest is

$$d_0^2 = \int_A (s_a - s_b)^2 = \int_A |s_a - s_b|,$$

The last simplification of the integrand is because the signals we are using are binary valued. The quantity d_0^2, which is equal to $d^2\,\sigma^2$, defines how the targets are "separated" in signal space. The larger this quantity is, the better one will be able to distinguish target s_a from target s_b.

Obviously d_0^2 depends on the characteristics of the two signals in question. We introduce the following notation. Let $d_0^2(n,\infty)$ to denote the value of d_0^2 when s_a = n-sided regular polygonal test pattern and s_b = Circle test pattern. Similarly let $d_0^2(n,I)$ denote the value of d_0^2 when s_a = n-sided regular polygonal test pattern and s_b = the image of the same -sided regular polygonal test pattern as obtained through imager I. Notice that $d_0^2(n,I)$ is simply a different name for the shape distortion measure $A_{rect}(n,I)$ which was defined previously.

Suppose that for two pairs of undegraded targets we get the following values (the units are immaterial)

$d_0^2(6,\infty)$ = 10000 for s_a = Hexagon test pattern, s_b = Circle test pattern

$d_0^2(4,\infty)$ = 20000 for s_a = Square test pattern, Circle, s_b = Circle test pattern

This says of course that it should be easier to distinguish between the Square and the Circle than to distinguish between the Hexagon and Circle. Now suppose that we pass the Hexagon and Square test patterns through some portion of the imaging system A, then segment the resulting image and compute

$d_0^2(6,A)$ = 10000 for s_a = Hexagon , s_b = Hexagon distorted by system A

$d_0^2(4,A)$ = 10000 for s_a = Square , s_b = Square distorted by system A

$d_0^2(6,A)$ is roughly the same value as the distance between the Hexagon and the Circle. If we know we are supposed to be looking for a Hexagon, we may try to set the threshold in the Matched Filter processor low enough that the distorted Hexagon pattern would pass the threshold and be called a Hexagon by the Matched Filter processor. But using this threshold, the Matched Filter processor would also declare that an ideal Circle is a Hexagon. Thus such a system would generate an error if its task is to distinguish between a hexagon and a circle. However, since $d_0^2(4,A) < d_0^2(4,\infty)$ it is possible to set a threshold to be able to distinguish between the distorted Square and the ideal Circle targets. Thus such a system would <u>not</u> generate an error if its task is to distinguish between a Square and a Circle.

Now say another imaging system B gives $d_0^2(6,B) = 20000$ and $d_0^2(4,A) = 20000$. In this case it is not possible to set a threshold so that either the Hexagon or the Square could be distinguished from the Circle. In this comparison, one would be able to say qualitatively that system A is better than system B since $d_0^2(n,A) \leq d_0^2(n,B)$ for the test patterns being considered, i.e. for the values of n considered.

The above discussion leads to the following method of characterizing the performance of an imaging system.

- Compute the Normalized Distortion Measure

$$\mathcal{M}(n,I) = \frac{d_0^2(n,I)}{d_0^2(n,\infty)} = \frac{\int_A |s_n - s_n^*(I)|}{\int_A |s_n - s_\infty|}$$

for s_∞ = ideal Circle, s_n = ideal n-sided regular polygon for n=3,..., and $s_n^*(I)$ = segmented result of passing an n-sided regular polygon shape through the subject imager (or portion of the imager).

- Find the value of n such that the Normalized Distortion Measure $\mathcal{M}(n,I) \approx 1$.

- Quote this value of n as the Maximum Resolvable Polygon (MRP).

Figure 4 describes the computation of the Normalized Distortion Measure $\mathcal{M}(n,I)$.

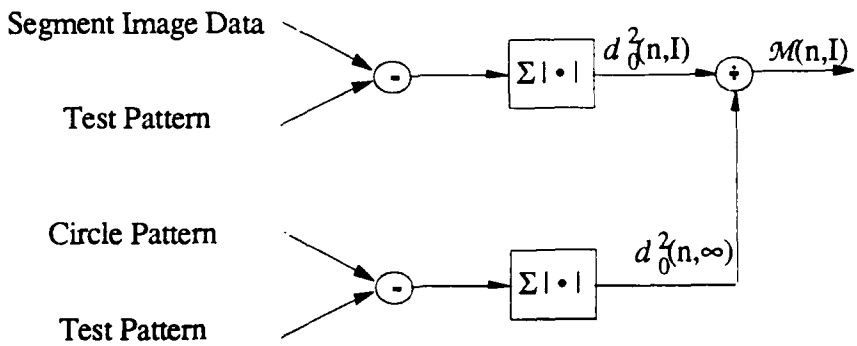


Figure 4. Computation of Normalized Distortion Measure $\mathcal{M}(n,I)$

The ratio $\mathcal{M}(n,I)$ can be plotted as a function of n. Better imaging systems will have lower values of $\mathcal{M}(n,I)$. Alternatively stated, better imagers have larger values of the MRP. They permit a Matched Filter to solve harder recognition problems because they introduce less shape distortion into the image data.

However, the MRP is more than just a qualitative measure of system performance. Like the MRT, the MRP defines the target situation (in the MRP case: the target shapes) for which the "probability of error" is barely equal to one-half. If the imaging system is presented with a regular polygon having n < MRP sides, then $\mathcal{M}(n,I) < 1$ which means that there exists a threshold which would

allow the Matched Filter processor to always distinguish between the polygon and the Circle shapes. Presenting a polygon with n > MRP sides means $\mathcal{M}(n,I) > 1$ and no threshold could be found which would give better results than guessing.

3.5. Noisy Imagers and Segmentation

Although the MRP criterion was developed above on the basis of noise-free imaging, it may also be used in cases where noise is present in the system. In the case of noisy imagers, the additive noise source may be viewed as a separate component of the imaging system which introduces some additional distortion on the test pattern.

There are two issues which must be resolved when admitting noise as one of the imaging system components. First is the issue of how to segment the image data. Recall that in order to compute $d_0^2(n,I)$, the image data must be segmented so that the targets pixels have value 1 and the background pixels have value 0. Segmentation in the absence of noise is easy since a simple threshold operator can separate background and target pixels in such a case. The situation is much more complicated when noise is present. Figure 5a shows a 5-sided polygonal target which has been imaged through a noisy sensor. Figure 5b shows the results of simple thresholding. The target area has holes in it and the background contains isolated pixels which have been classified as target pixels. This simple result is not appropriate for shape distortion analysis. Most reasonably good ATR systems would be able to segment the target from the background better than this. What is needed is a baseline segmentation algorithm that is equivalent to thresholding when no noise is present, but which clearly isolates the target pixels when noise is present. We have settled on a two step segmentation procedure:

- Threshold the image at a value midway between the original target and background intensity values.
- Fix-up the resulting binarized image by:
  - (i) Filling in the holes in the target area
  - (ii) Discarding the incorrectly classified background pixels

The following is a description of our first attempt at an algorithm to perform these two "fix-up" tasks:

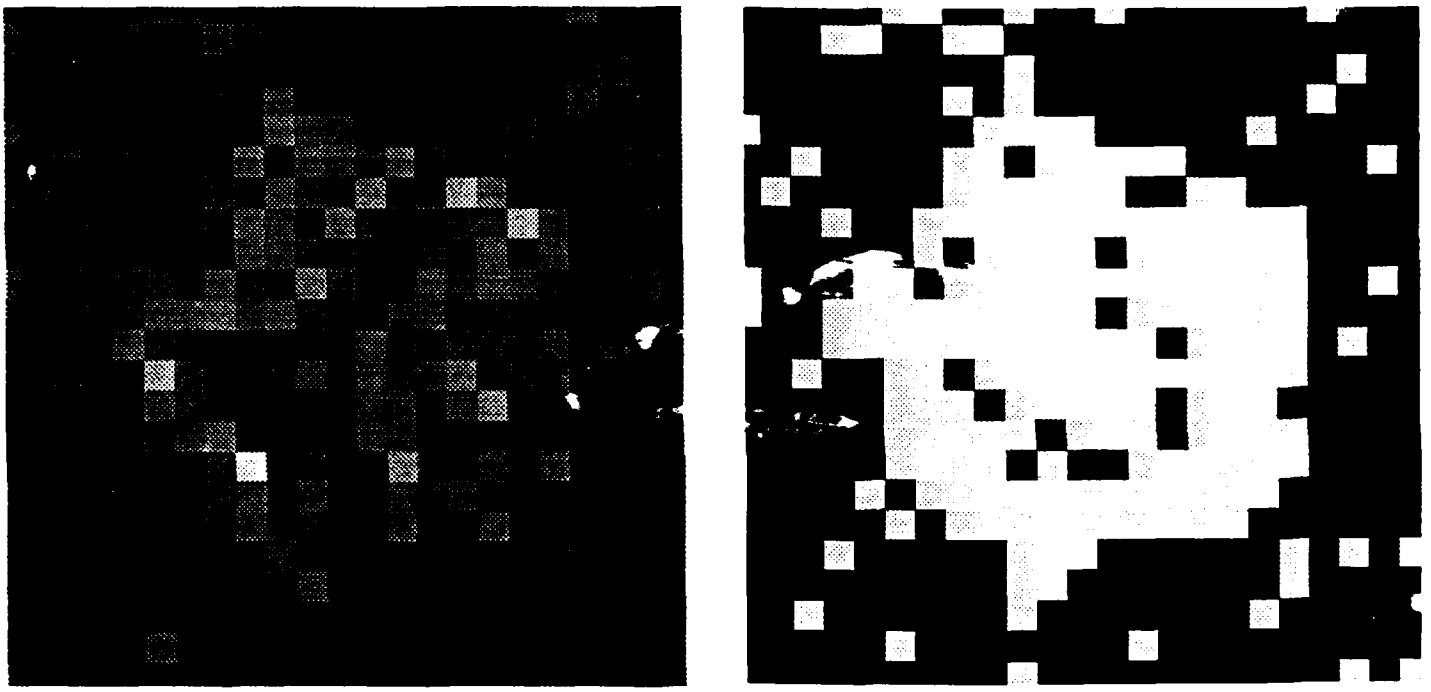
Algorithm A to "Fix-Up" the Thresholded Image

(a) Detect edge pixels in the thresholded image.

(b) Determine the size of each region which is enclosed by the edge pixels.

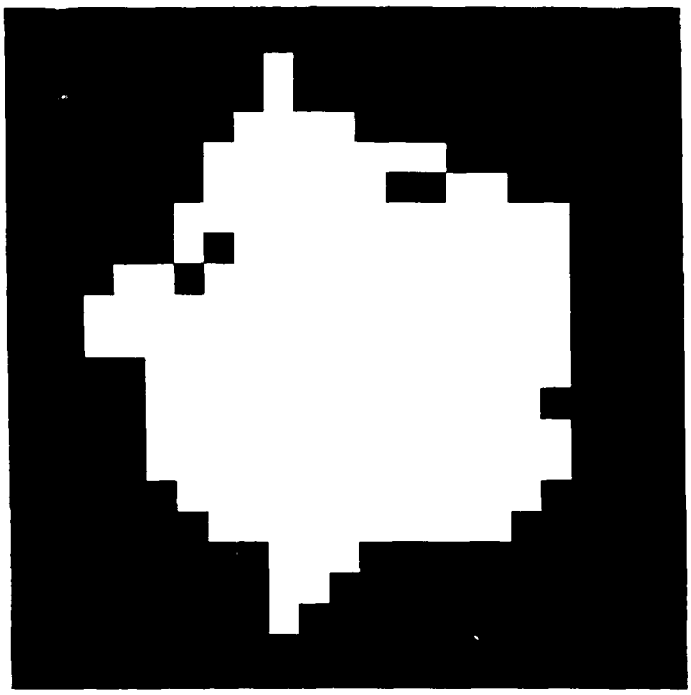
(c) Discard those edge pixels surrounding all regions except the largest region, which is assumed to be the target.

(d) Label all the pixels inside the one remaining region as target pixels.

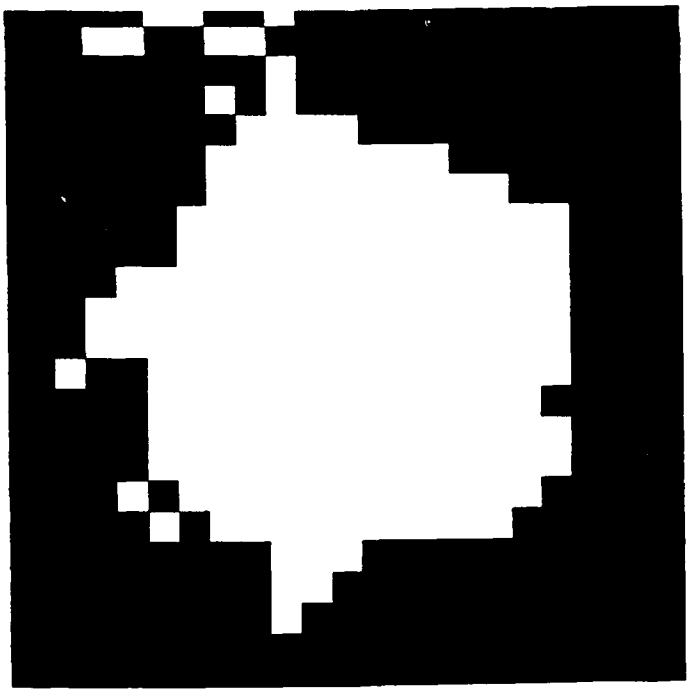


a)

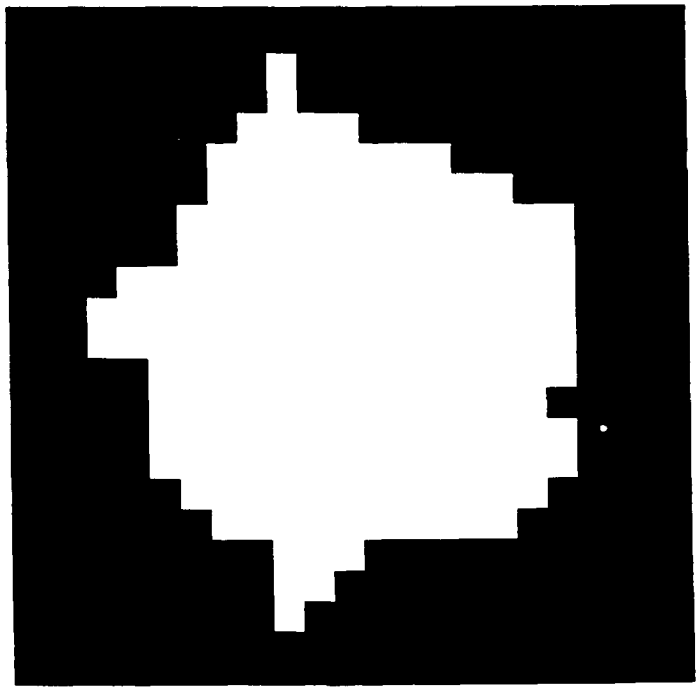
b)



c)



d)



e)

Figure 5. Segmenting a Noisy Image:
a) The noisy original image of a 5-sided polygon,
b) Result of thresholded the image in (a),
c) Fix-up by Algorithm A assuming the object in (b) is 4-connected,
d) Fix-up by Algorithm A assuming the object in (b) is 8-connected,
e) Result of the final Baseline Segmentation Algorithm on the noisy image

If Algorithm A is followed as stated, there are two possible results, which are determined by whether the object is assumed to be defined by 4-connectedness or 8-connectedness. (To understand the difference between 8-connected and 4-connected regions, see the Appendix). Figure 5c shows the results of performing the above algorithm manually assuming that the target object we are looking for is defined by 4-connectedness. This does not appear quite right since there are still some pixels which we would consider to be "background" embedded within what appears to be the "target". The problem is that in the process of finding the edges of an object which is assumed to be 4-connected, an 8-connected background is generated. Assuming the target to be 8-connected does not improve the situation as Figure 5d shows. Although the holes within the target region are gone, the 8-connected target assumption attaches too many pixels tot he target which in reality appear to be barely connected to the real target.

In consideration of the above discussion, we now define a segmentation algorithm which appears to overcome the stated difficulties. By adding one additional step to the above algorithm, we are able to segment objects so that both the object and the background appear to be 4-connected objects.

Baseline Segmentation Algorithm

1. Threshold the image at a value midway between the original target and background intensity values.

2. Fix-up the resulting binarized image by:

2.1 Examine each pixel which is labeled as a target pixel. If it is not 4-connected to another target pixel, then re-label the pixel as background.

2.2 Detect edge pixels of all 4-connected objects.

2.3 Determine the size of each 4-connected object which is enclosed by the edge pixels.

2.4 Discard those edge pixels surrounding all objects except the largest one, which is assumed to be the target.

2.5 Label all the pixels inside the one remaining object as target pixels.

Figure 5e shows the result of applying this algorithm to the noisy image of Figure 5a. Although the software implementation of this algorithm is straightforward, we have yet written the code to perform it. Therefore, in the experiments reported on below, we performed this algorithm manually.

The second issue which must be resolved is the fact that noise introduces randomness. Since each noise realization is slightly different, the exact distortion introduced by the noise differs from trial to trial, which makes the MRP seem to be a random quantity. What is desired is the average MRP, which means that the average value of $d_0^2(n,I)$ must be computed. This implies a Monte Carlo collection of simulations. However, since $d_0^2(n,I)$ is itself computed from an integral over the effects of the noise on the individual pixels, in fact only very few Monte Carlo runs need to be performed to get an accurate estimate of the mean of $d_0^2(n,I)$.

4. Description of the Performance Modeling Software

In this phase of the research we have concentrated on developing a software tool for the analysis of the imaging sensor subsystem only. This software tool is described in this section. Future work will include software hooks for analyzing the algorithm subsystem as well. Plans for how to accomplish this extension are given in the section of this report entitled "Feasibility of the Method". All the software was developed on a Macintosh II computer.

4.1. Control Software Environment

Each of the imaging system components is simulated in a separate FORTRAN subroutine. Listings of the source code for these subroutines are given in the Appendix. To create an overall simulation of the imaging system requires that each subroutine be called in turn. While this can obviously be done from a FORTRAN control driver routine, we have taken a more interactive approach which allows us to stop the simulation in progress, view partial results, continue the simulation, etc. In this approach, a commercial image processing software package, called *IPLab*, which is made and marketed by Signal Analytics is used to

- Control the simulation of the imaging system
- Obtain parameter inputs from the user
- Control the display of image data
- Maintain data files associated with the simulation
- Perform the shape distortion computations.

The imaging system modeling subroutines and some *IPLab* control functions are grouped in a sequence called a Script in *IPLab*. A Script is essentially a mini-sequence controller which is easily created and edited from within *IPLab* and it may be stopped at any time to view partial results. A listing of the commands in one such *IPLab* Script which simply makes a high resolution version of the data is given below.

***IPLab* Command Name**	**Comment**
Pause	:***Setup
Open	:Get input parameters from vector
Select All	:Paste into IPLab Variables
Copy	:
Dispose Window	:
Show Variables	:
Select All	:
Paste	:
UserProcedure1	:Read input parameters from Variables
Pause	:***Make the test pattern
New	:Put it in a new window
UserProcedure2	:Fill the 512x512 array
Nonlinear Filter	:Erode to smooth rough edges
Nonlinear Filter	:Erode to smooth rough edges
Pause	:***Multiply by MTFs
Select All	:

```
Transform              :FFT the scene data
UserProcedure3         :Apply Optical MTF to Magnitude FFT
UserProcedure4         :Apply Detector MTF to Magnitude FFT
Transform              :Inverse FFT
Change Data Type       :Make it Integer data type
Rename Window          :Call it Data
Change Window          :Dispose of imaginary part
Dispose                :     "
Pause                  :***Apply Sampling and Noise
UserProcedure5         :Sampling and noise
Show Display           :Display image
END                    :Return interactive control
```

The commands labeled UserProcedure1 - UserProcedure5 designate FORTRAN subroutines which have been written specifically to implement portions of the imaging system simulation.

The software accepts inputs into a vector which can be saved for repeated use. The input parameters are as follows:

nSides	=	Number of sides in the polygonal target
tDiameter	=	Diameter of circular target (meters)
tRange	=	Range to target (meters)
orient	=	Rotation angle for target or clutter
TargOrClut	=	0 to generate a target shape 1 to generate clutter shape
amplitude	=	Amplitude of random variable added to polygonal shape to make clutter
bandwidth	=	Bandwidth of random variable added to polygonal shape to make clutter
opticalF0	=	Optical cutoff freq f_0 = optical diameter/wavelength.
IFOVdetx	=	detector IFOV in x-direction (in mr)
IFOVdety	=	detector IFOV in y-direction (in mr)
rectOrHex	=	0 for rectangular sampling, 1 for hexagonal sampling
SNR	=	RMS signal to noise ratio.Must be > 0.
seed	=	For random number generator

4.2. Imaging System Simulation

Sensor resolution is affected by image blurring (due to the optics, the detector size and shape, and blur due to motion of the camera or objects in the scene) and spatial sampling. There are two types of noise introduced by the sensor system: spatial, which can be seen in a single frame of image data, and temporal, which can only be seen frame to frame. We consider only single frame processing, so temporal noise is ignored for the moment. The sum of the various independent noise sources in a single image frame is often usefully modeled as a spatially stationary Gaussian random process, which is generally characterized by its variance and correlation function, or equivalently, the noise power spectrum. Detector non-uniformity is a non-zero mean noise source which cannot be modeled as a spatially stationary random process, so it is usually handled separately. The atmosphere contributes both resolution degradation, which is similar to the optical effects, and signal attenuation, which lowers signal strength and hence the signal to noise ratio. Distortion at the output of the detection stage is naturally represented in terms of deviations from the pixel intensities which would be obtained from a perfect imaging system.

The following system components have been simulated in the current version of our performance modeling software:

- System inputs: Generation of synthetic test patterns
- Optical MTF
- Detector MTF
- Sampling due to the detector array geometry
- Additive noise

The model formulations for each of these components is given next.

4.2.1. Test Pattern Generation

The test patterns are generated in an array of size 512×512 samples of integer data type. There is only one test pattern object in the array, and it is centered at location (256,256). The background is taken to have value 0 and the interior of the target (or clutter) object is filled with value 200. All simulations of the imaging system are performed on this array or the Discrete Fourier Transform of the data in this array.

The polar coordinate representation formulas given in the section above on The Test Pattern Set are used to define the boundaries of the test patterns

$$\rho(\theta,\infty) = r(\infty), \qquad \text{for a circle}$$

$$\rho(\theta,n) = r(n)/\cos(\,(\theta \bmod (2\pi/n)\,) - 2\pi/2n\,) \qquad \text{for a polygon with n sides}$$

$$\rho_{clutter}(\theta,n) = \rho(\theta,n) + U(\theta,a,b) \qquad \text{for a clutter object}$$

where the random function U is obtained by filtering the outputs of a random number generator called ranu

$$U(\theta_k,a,b) = (1-b)*U(\theta_{k-1},a,b) + b*\text{ranu}(k)$$

To generate the test patterns the angle parameter $0 \le \theta \le 2\pi$ is sampled into 1000 elementary units

$$\theta_k = 2\pi k/1000, \qquad k = 0,1,...,999.$$

The radius r of the circle is taken to be

$$r(\infty) = 180 \quad \text{for the circular target.}$$

The parameter r is adjusted for each of the other target test patterns so that the average of the maximum and minimum radii over the target is equal to 180. This requires that

$$r(n) = \frac{2 \cdot 180}{1 + 1/\cos(\pi/n)} \qquad \text{for a polygon with n sides}$$

4.2.2. Optical MTF

Diffraction limited circular disk optics are assumed. The MTF is

$$H(\Omega) = \frac{2}{\pi} \left\{ \cos^{-1}(\Omega) - \Omega(1 - \Omega^2)^{1/2} \right\}$$

where the normalized radial frequency Ω is given by

$$\Omega = \frac{(f_x^2 + f_y^2)^{1/2}}{f_0} \text{ (cycles/mr)}$$

$$f_0 = \frac{\ell}{\lambda f_\#}$$

λ = wavelength (micrometers)

$f_\#$ = optical f-number

ℓ = focal length of the optical system (micrometers)

Notice that the two-dimensional MTF is used in the Shape Distortion Analysis software; a one-dimensional approximation is not required as it is in the MRT analysis. To simplify input, the software accepts only f_0.

4.2.3. Detector MTF

Rectangular shaped detectors are assumed. The analog MTF of such a detector is

$$H_{analog}(f_x,f_y) = \frac{\sin(\pi D_x f_x)\, \sin(\pi D_y f_y)}{D_x f_x D_y f_y}$$

where D_x and D_y are the horizontal and vertical dimensions of the detector. In a digital computer simulation where analog lengths must be simulated by some number of digital samples, the following formula gives a more accurate representation of the desired MTF

$$H_{digital}(f_x,f_y) = \frac{\sin(\pi N_x f_x)\, \sin(\pi N_y f_y)}{N_x \sin(f_x) N_y \sin(f_y)}$$

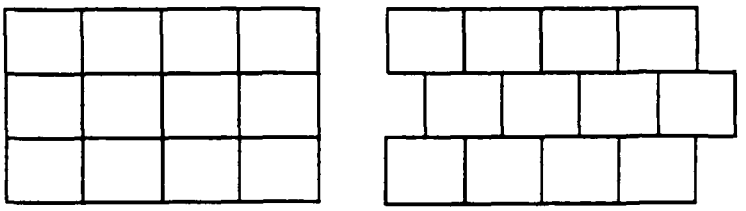
where N_x is the number of samples in the simulation required to represent D_x, and similarly for N_y. The inverse DFT of $H_{digital}$ is indeed a rectangular pulse, whereas the inverse DFT of the H_{analog} demonstrates undesired oscillations. $H_{digital}$ is used in the software simulation.

4.2.4. Sampling and Noise

The performance modeling software allows for two different types of sampling imagers:

- Rectangular detectors centered on a rectangular lattice
- Rectangular detectors centered on a hexagonal lattice

The difference between these two types of detectors is shown in Figure 6. The hexagonal detector array has alternate rows displaced by one-half pixel width. This detector array geometry was chosen over the ideal hexagonal close-packed detector array geometry (in which each detector element is a true hexagon) due to its simplicity of simulation. However, this is a reasonable first approximation to hexagonal close-packed detector elements. Furthermore, this detector array geometry is also of interest because the technology for manufacturing detector arrays forces one to consider rectangular detector elements before considering the more complicated hexagonal detector elements.



a) Rectangular Lattice b) Hexagonal Lattice

Figure 6. Geometry of Rectangular and Hexagonal Lattices of Rectangular Detectors

Our objective in simulating the degrading effects of an imaging system is to obtain an image of a target whose shape can be compared to the original. Since sampling reduces the size of an object, the sampled image itself cannot be directly compared to the original without expanding it to the same size as the original. This re-sizing implies some sort of interpolation must be done. We use zeroth order interpolation, i.e. simple pixel replication, for both the rectangular and hexagonal sampling arrays.

The additive noise is modeled as white (i.e. spatially independent) and Gaussian distributed. It is added to each pixel as the sample is formed.

4.3. Shape Distortion Computation

The Normalized Distortion Measure $\mathcal{M}(n,I)$, as defined in Figure 4, requires performing the image segmentation, taking the absolute value of the difference between certain images, summing the pixel values in the absolute difference images, and computing a ratio. All these operations except the segmentation step are performed in an *IPLab* Script. The segmentation required in the computation of $\mathcal{M}(n,I)$ is accomplished in this version of the software via simple thresholding:

$$\text{data}(i,j) = \begin{cases} 1 & \text{if data}(i\ j) \geq \text{target value} \\ 0 & \text{otherwise} \end{cases}$$

When noise is present the algorithm described in Section 3.5 for "cleaning" the thresholded image is performed manually.

5. Experimental Results

In order to test the validity of the Shape Distortion Analyzer we must compare the performance as predicted by it to the actual performance of some target recognition subsystem. Since the handiest target recognition system is human vision, we decided to compare the computed ranked performance of several subject imaging systems with the ranked performance of the same simulated imaging systems as perceived by humans. In order to make the human recognizer more

like an automatic target recognizer, we only present the segmented images to the human. This eliminates the ability of the human visual system to extract additional information from the grayscales images in any nonlinear (read: "Not currently understood") way in order to resolve shapes. It must also be noted that no specific attempt has been made to design an accurate psycho-visual testing environment. In fact one may expect biases in this initial set of experiments. However, the results at least give a preliminary comparison of the use of the MRP vs. human recognition.

In general, we use the following experimental procedure:

(i) The portion of the imaging system which is to be studied is selected. This may include any combination of: (a) optical MTF; (b) detector MTF; (c) sampling; (d) additive noise.

(ii) An *IPLab* script is constructed which includes those selected components of the optical system in the portion of the Script which simulates the imaging system.

(iii) The rest of the *IPLab* Script is setup to compute the Normalized Distortion Function $\mathcal{M}(n,I)$ and loop over n, the number of sides in the test polygons.

(iv) As the image of each polygonal test target is segmented for computing $\mathcal{M}(n,I)$ it is also printed on a laser printer.

(v) The value of n at which $\mathcal{M}(n,I)$ crosses unity is noted as the Maximum Resolvable Polygon.

(vi) The human subject is shown the same series of images and in each case is asked to decide whether the image looks more like it came from a circle or a regular polygon of the given number of sides The value of n at which this choice becomes ambiguous is noted as N_{human}, the 50% recognition point for humans.

Only some of the front end imaging system components have been studied. Satisfactory results from these experiments would justify extending the analysis to include other components of the imaging system, such as atmospheric effects and the algorithmic sections of the system itself.

Two different studies were performed:

- Hexagonal sampling versus rectangular sampling
- The effect of noise

Certain nominal parameter values were used in the experiments. The values of the parameters marked with an astcrisk (*) below are taken from the test case which appears in the NVL Static Performance Model Report. Other parameters represent our best judgement regarding useful ranges of values.

nSides	=	3 to 10
tDiameter	=	10 meters (roughly the size of a tank)
tRange	=	2000 meters
orient	=	0
TargOrClut	=	0 (only target shapes were used in the experiments reported on here)
amplitude	=	0
bandwidth	=	0
*opticalF0	=	10000 (corresponds to optical diameter=4 in., wavelength = 10μm)
*IFOVdetx	=	.25 mr

*IFOVdety	=	.25 mr (nominal for rectangular sampling lattice)
rectOrHex	=	0 for rectangular sampling, 1 for hexagonal sampling
*SNR	=	1000 down to 2.5 (2.5 is the NVLSPM test case value)
seed	=	0

5.1. Hexagonal Sampling vs Rectangular Sampling

First we give a word of caution about the results presented here on comparing hexagonal versus rectangular sampling lattice detector geometries. This study only analyzes shape distortion by the imaging sensor. No account is taken of the relative merits of hexagonal versus rectangular lattices in algorithm performance. The results here show there is little difference in the shape distortion between the two types of sensors. This means that the quality of the image data passed to the algorithm subsystem is virtually identical for either sensor. If this result is borne out by further experimentation, it means that imaging system designers would be free to choose the lattice configuration which provides the best algorithm performance. For example, there is significant evidence that hexagonal sampling yields images which are "more recognizable" by algorithms based on mathematical morphology.[3] With the growing interest in and success of pattern recognition algorithms based on mathematical morphology, researchers in that field should be happy to find that the shape data they could get from a hexagonal sampling lattice is just as good as that which may be obtained from a rectangular sampling lattice.

We have already discussed the use of rectangular sampling elements for both the rectangular and hexagonal sampling arrays in this simulation. One must also consider the inter-sample spacing. For this experiment, the nominal parameter values were used for the rectangular sampling lattice, so the detector IFOV is .25 mr square. However, for the hexagonal lattice, refer to Figure 7. In a true hexagonal sampling lattice, the center-to-center distances between adjacent detectors is the same, so the spacing between rows of detectors is less than the horizontal sample spacing. We want to know whether this reduced vertical sample spacing helps the imager performance, so we have compared three kinds of samplers:

- Rectangular lattice with .25 mr square IFOV (Rect Imager)
- Hexagonal lattice with .25 mr square IFOV (Hex-Square Imager)
- Hexagonal lattice with horizontal IFOV = .25 mr and vertical IFOV = $\frac{\sqrt{3}}{2}$ (.25) = .22 mr (Hex-Nonsquare)

In this experiment, the detectors are rectangular and contiguous in each of the sampling arrays.

---

[3] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press, 1982. See Chapter VI on Digital Morphology for a discussion of the relative merits of rectangular versus hexagonal sampling with respect to morphological algorithms.

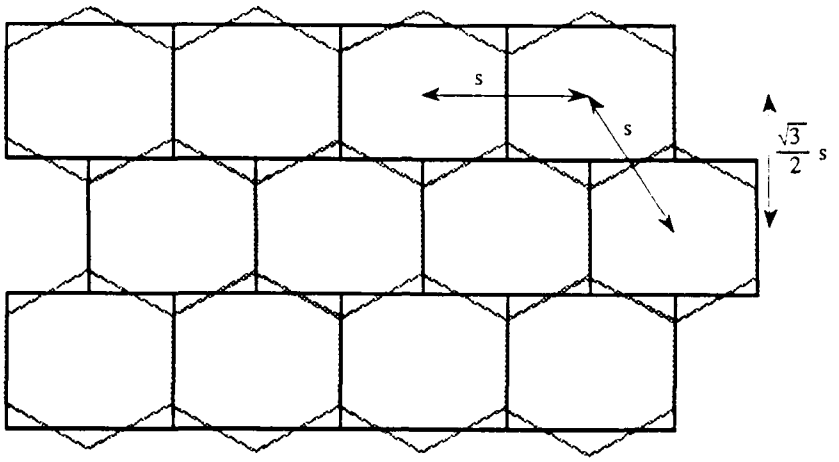


Figure 7. Detector Spacing in a Hexagonal Lattice

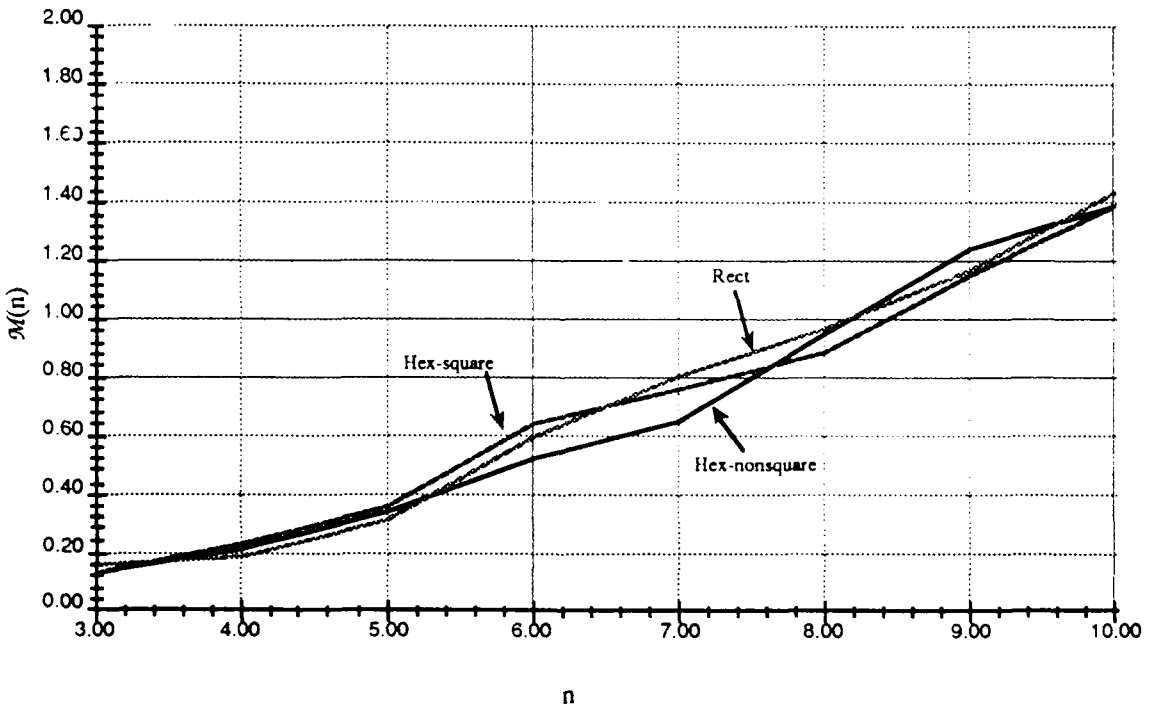


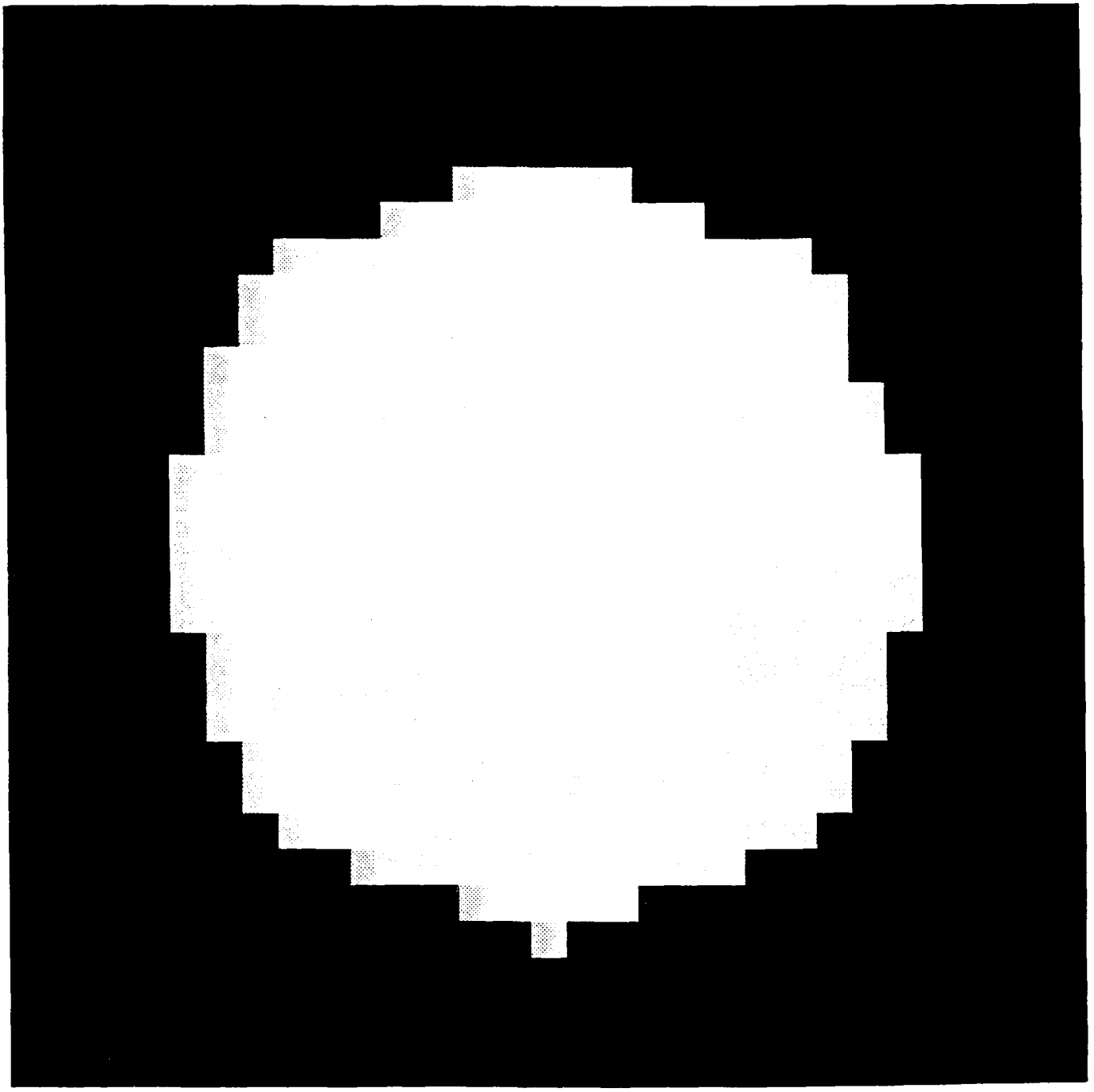
Figure 8. Normalized Distortion Measure for Three Imagers

The computed Normalized Distortion Measure $\mathcal{M}(n,I)$ is plotted in Figure 8 for each of these three imagers. The MRP and N_{human} are summarized in Table 1 below. The imaged polygons with N_{human} and N_{human} +1 sides for each imager are shown in the Figure 9. Although by human recognition, the Hex-Nonsquare lattice performs slightly better than the other two imagers, this experiment suggests that hexagonal lattice samplers and rectangular lattice samplers degrade the image data fairly equally from the viewpoint of an automatic target recognizer. From visual inspection, the hexagonal lattice samplers can follow edges along certain orientations better than rectangular samplers. The notable exception to this is vertically oriented edges, which are poorly tracked by the hexagonal samplers (For example see the right edges of the 9-sided polygons in Figures 9c and 9e). Each of imagers produces artifacts; however, humans appear to be better able to reject the boundary irregularities produced by the hexagonal samplers than a Matched Filter can.

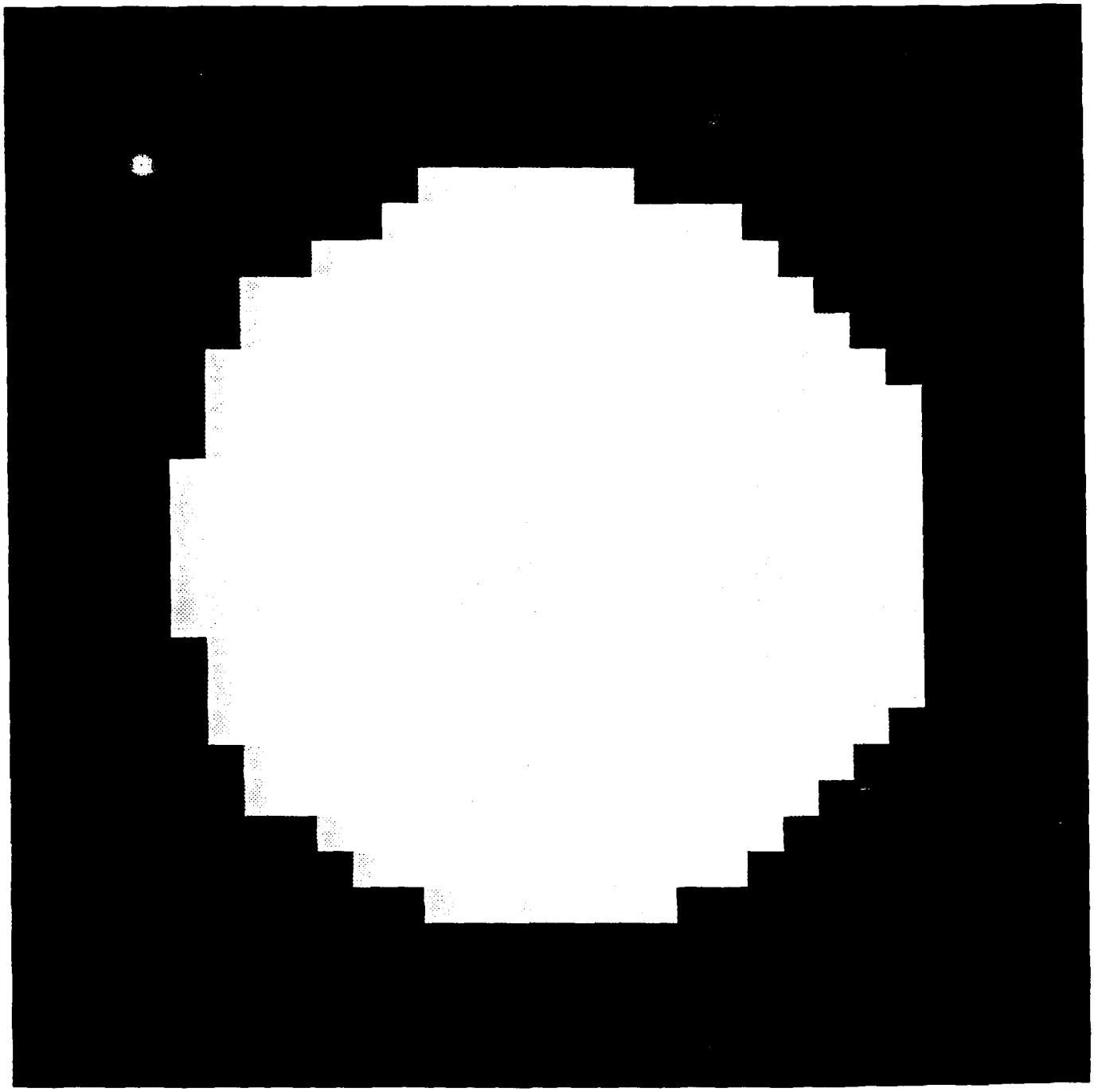
Table 1. Comparing Sampling Geometries

Imager	MRP	N_{human}
Rect	8.2	8-
Hex-Square	8.4	9-
Hex-Nonsquare	8.2	9+

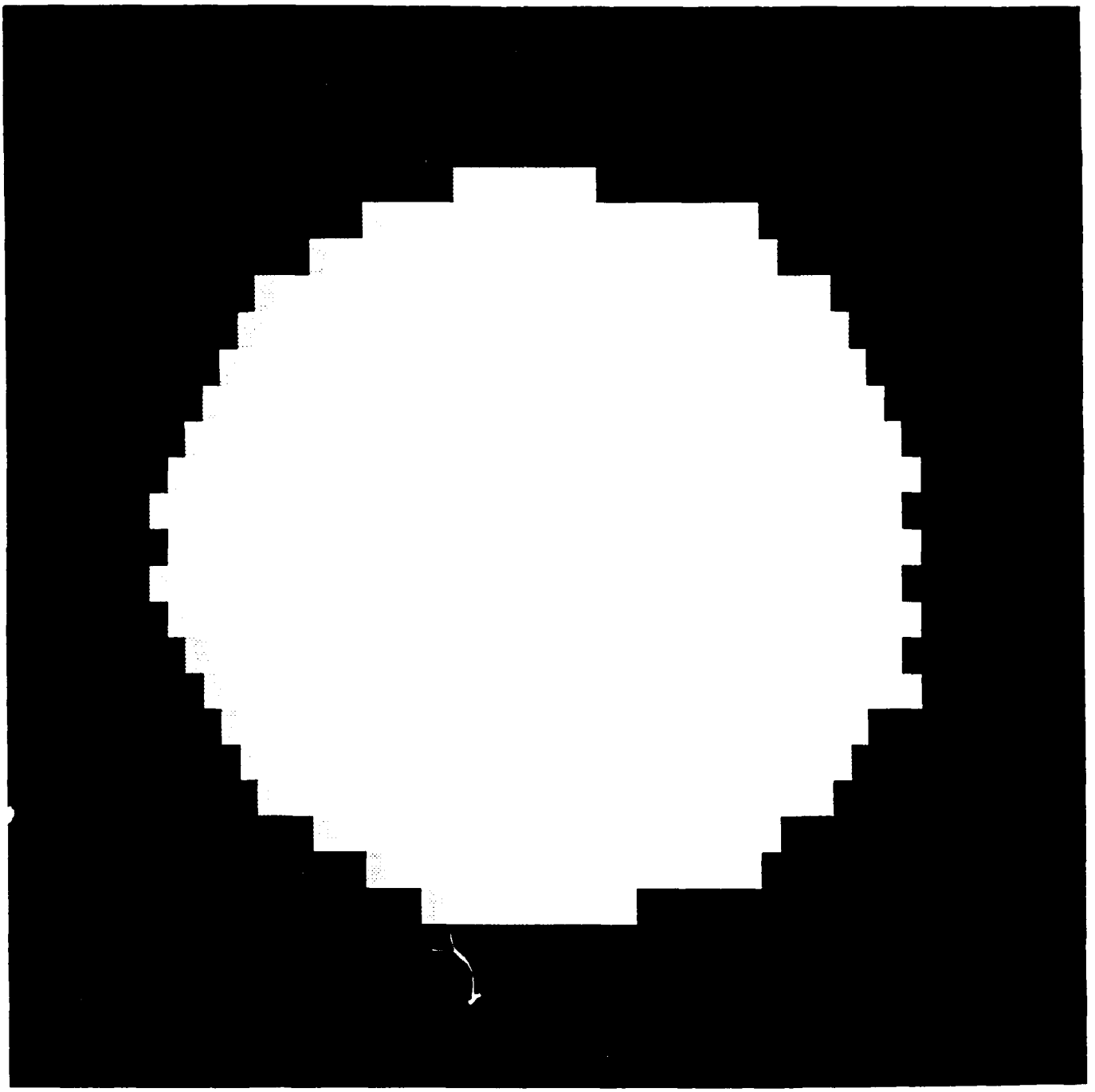
Notice that performance based on MRP does not really improve even when the smaller detector elements are used for the hexagonal sampler. This result seems to contradict the theoretical result that hexagonal close packed detector elements should sample the image more efficiently. However, keep in mind that the detector elements here were rectangular. The results from this experiment say only that you should not expect any performance gain from sampling on hexagonal centers if you take the easy way out and make your detectors rectangular. It is still uncertain what would happen if you use hexagonal detectors.



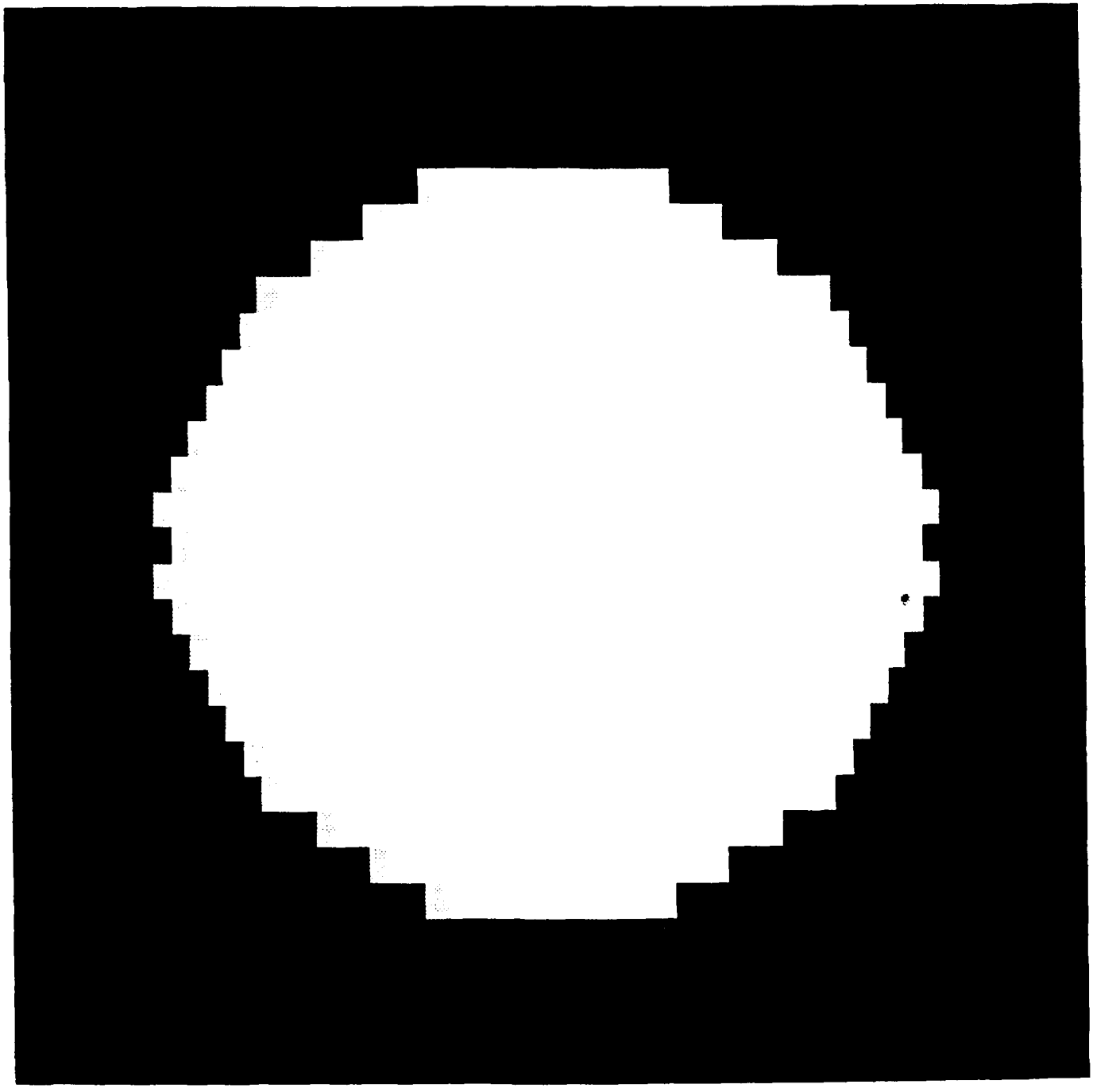
a) Rectangular lattice, 8-sided polygon.



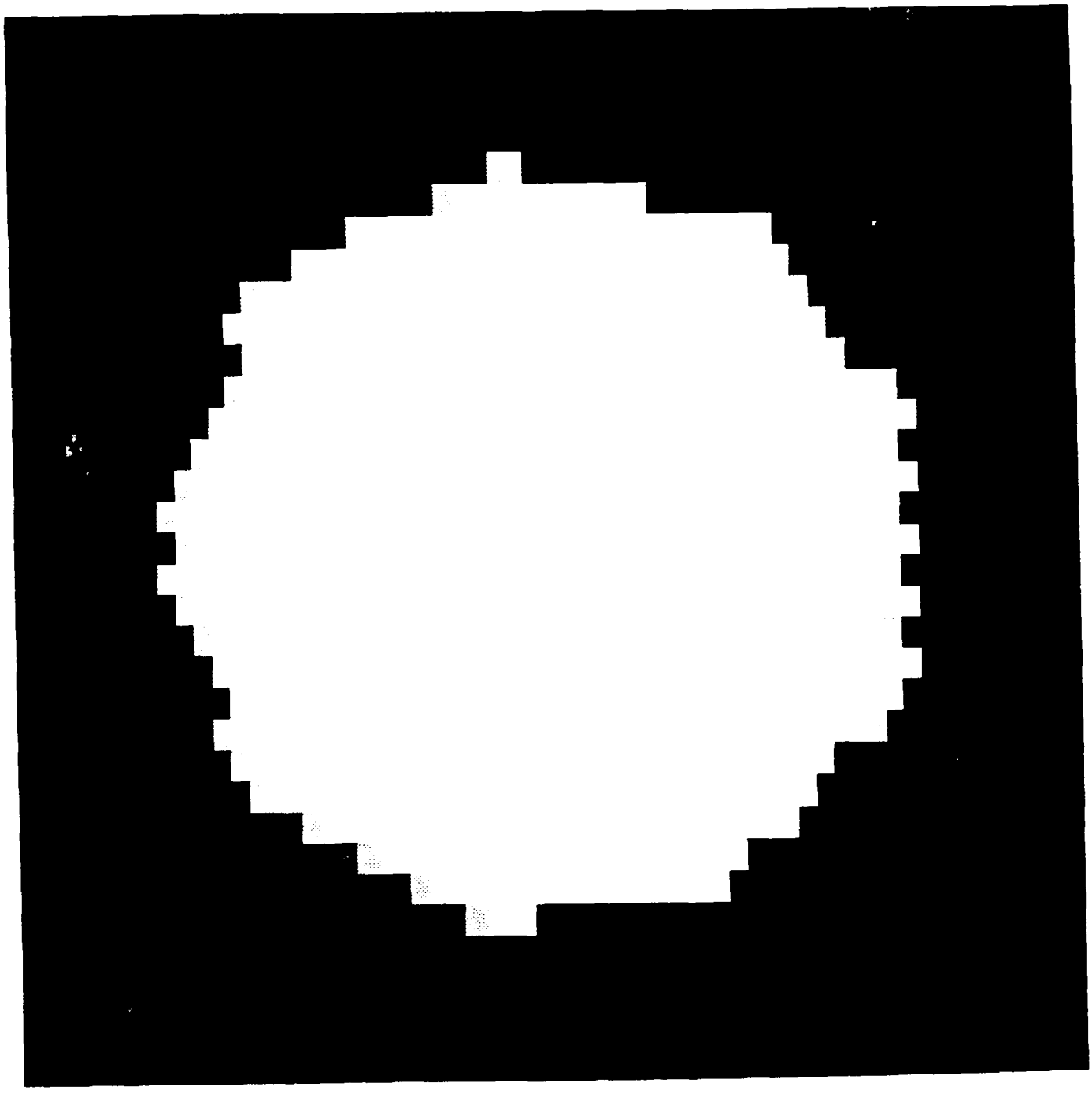
b) Rectangular lattice, 9-sided polygon.



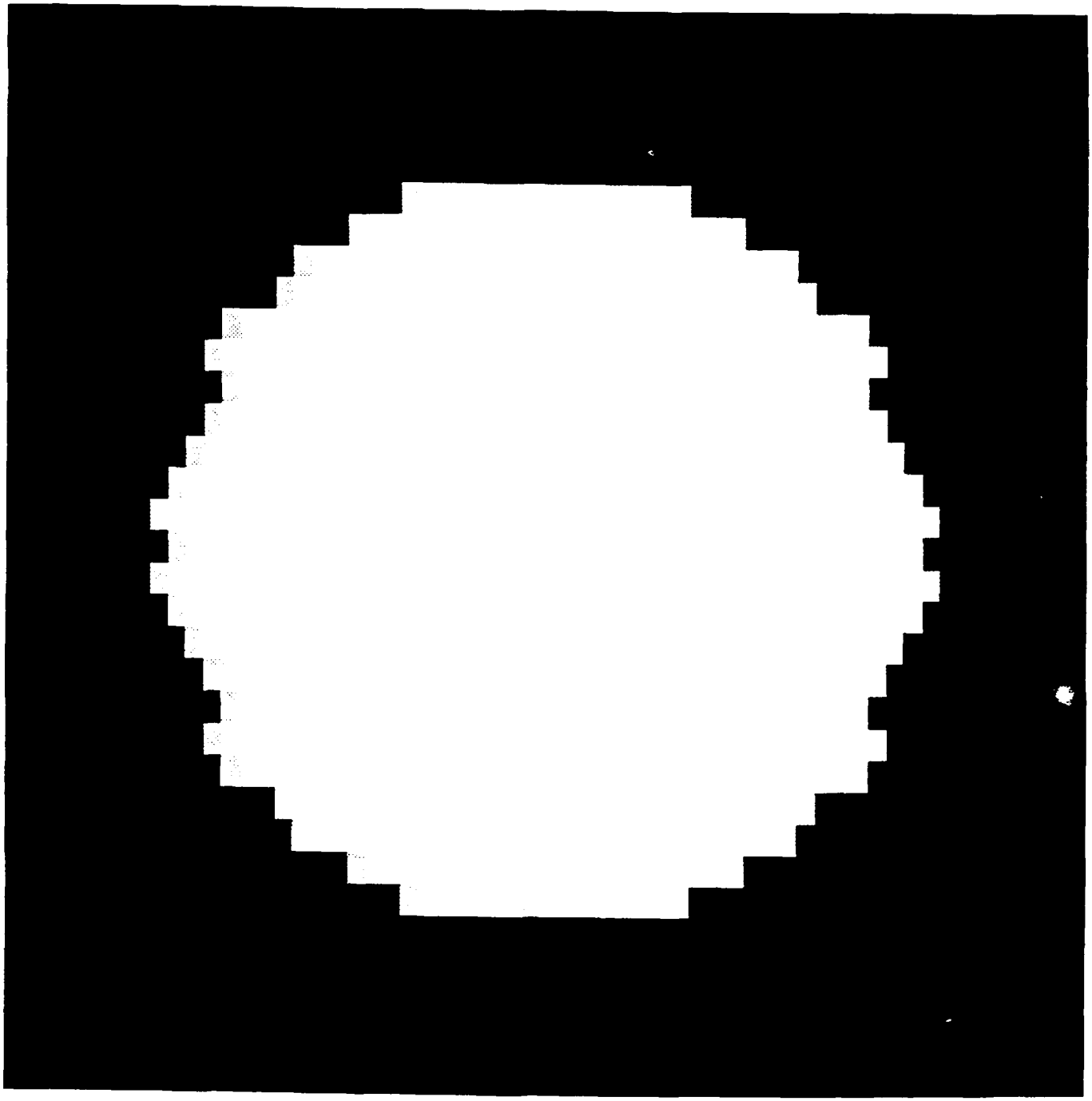
c) Hex-Square lattice, 9-sided polygon.



d) Hex-Square lattice, 10-sided polygon.



e) Hex-Nonsquare lattice, 9-sided polygon.



f) Hex-Nonsquare lattice, 10-sided polygon.

Figure 9. The Segmented Images for the Experiment Comparing Rectangular and Hexagonal Sampling

5.2. The Effect of Noise

The Normalized Distortion Function $\mathcal{M}(n,I)$ was computed for signal to noise ratios of 2.5 and 4.0 and the results are plotted in Figure 10 together with the results for an essentially infinite SNR.

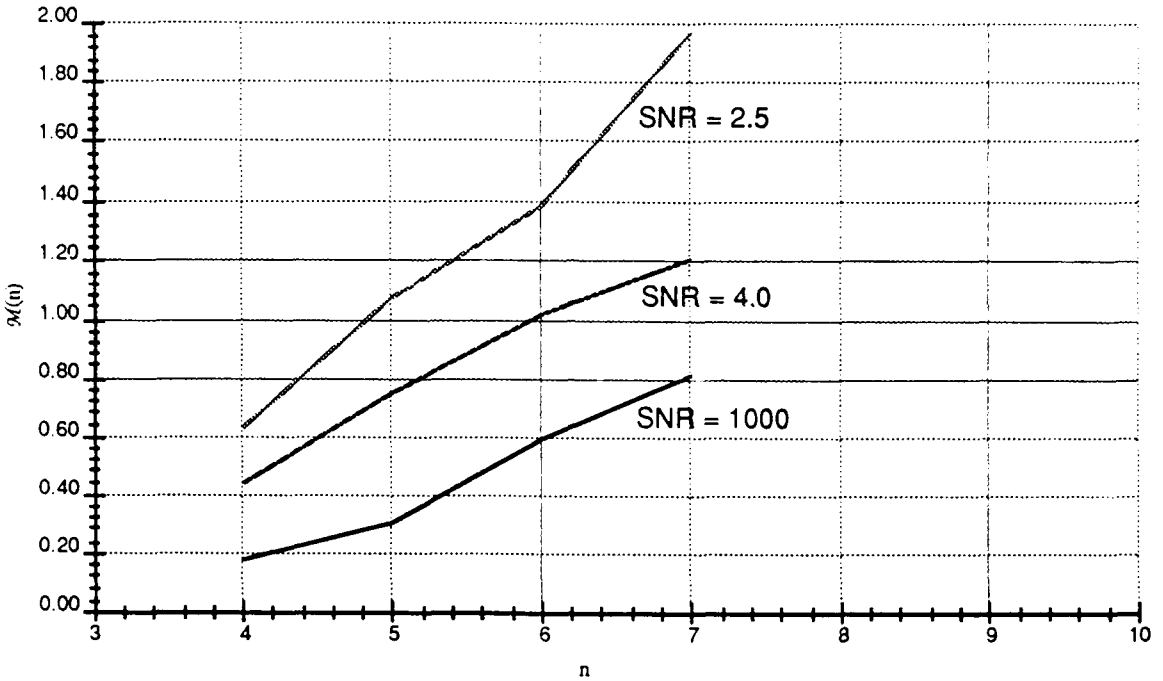


Figure 10. Normalized Distortion as a Function of n and SNR

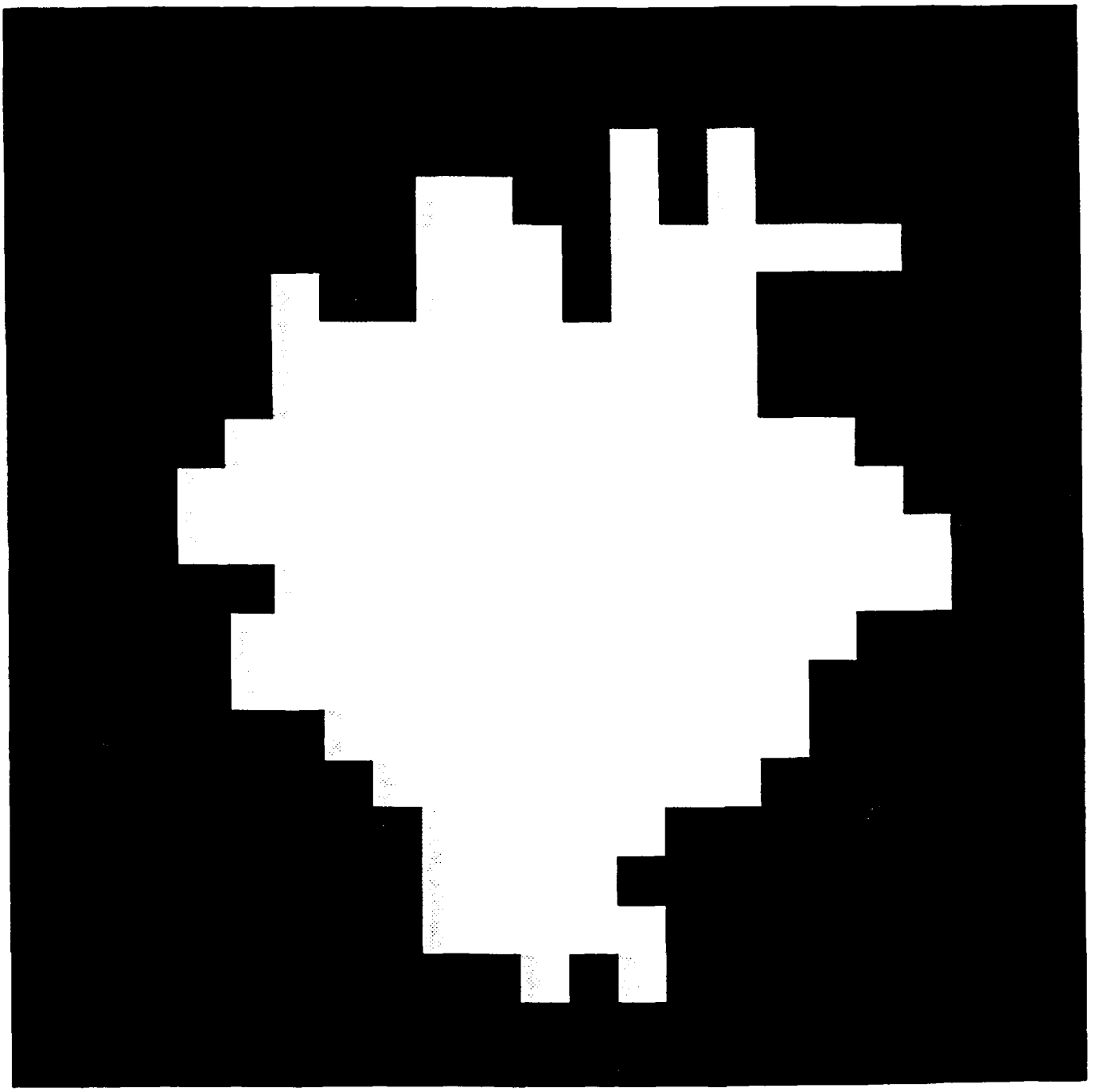
To compare the influence of noise on the measure $\mathcal{M}(n,I)$ to human performance, we have taken a slightly approach than in the previous experiment. We fixed the value of n, the number of sides in the regular polygon, and varied the signal to noise ratio to (i) the point at which $\mathcal{M}(n,I) \approx 1$ and (ii) the point at which a human barely can discern the object from a Circle. The results are summarized in Table 2 below.

Table 2. SNR Required for Recognition

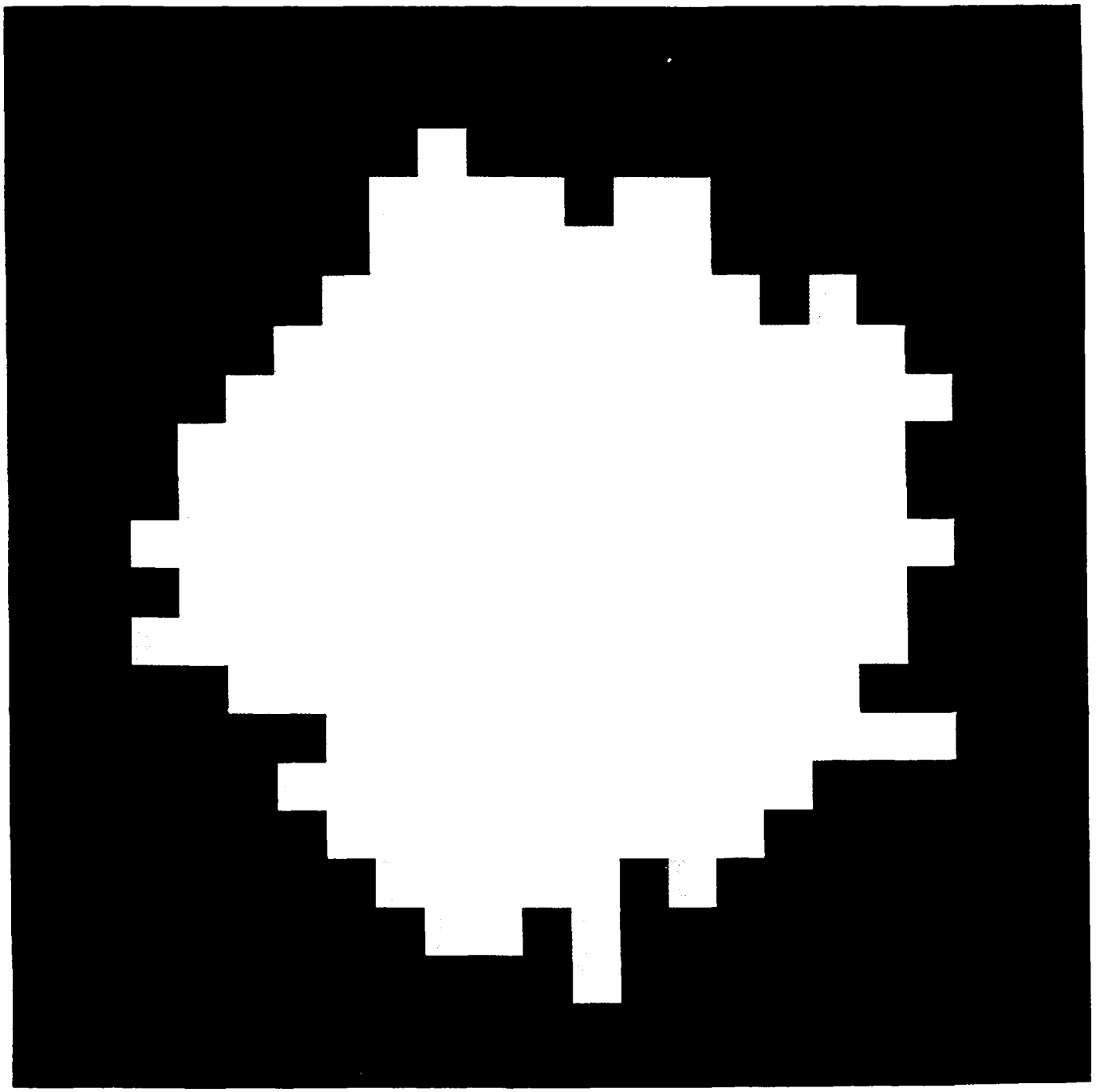
n	SNR at $\mathcal{M}(n,I) \approx 1$	SNR for human recognition
4	1.25	1.25
5	2.5	2.5
6	4.0	4.0
7	5.0	4.5

The hand segmented images for each of the human recognition cases are shown in Figure 11. The required SNR as predicted by the MRP is in quite close agreement with the observed performance of a human. Notice that the required SNR depends on the task. In performing the task of distinguishing between a square and a circle, both the human and the predicted automatic target recognizer can tolerate a fairly low signal to noise ratio. In fact the sqaure shape is quite distorted, but it can still be said with some confidence that it could not have come from a Circle target shape. However, since a hexagon is much closer to a circle to begin with, much less noise can be tolerated by both the human and the predicted automatic target recognizer for them to discern the object from a Circle.

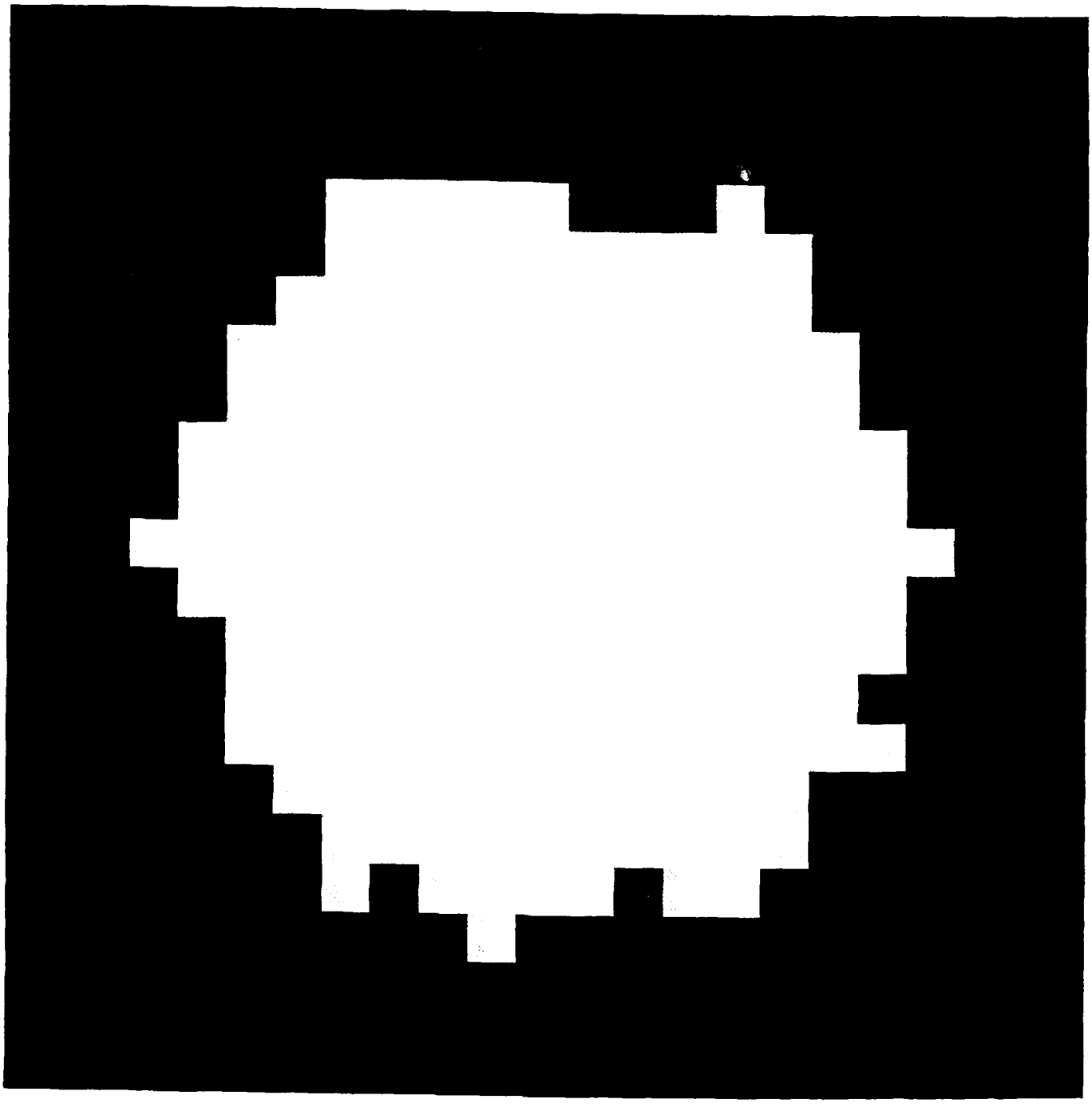
It should also be noticed that the noisy segmented shapes can be non-convex. This is one of the primary difficulties in using a polar coordinate representation of the image shapes. Since the polar coordinate shape representation could only be used to represent convex shapes.



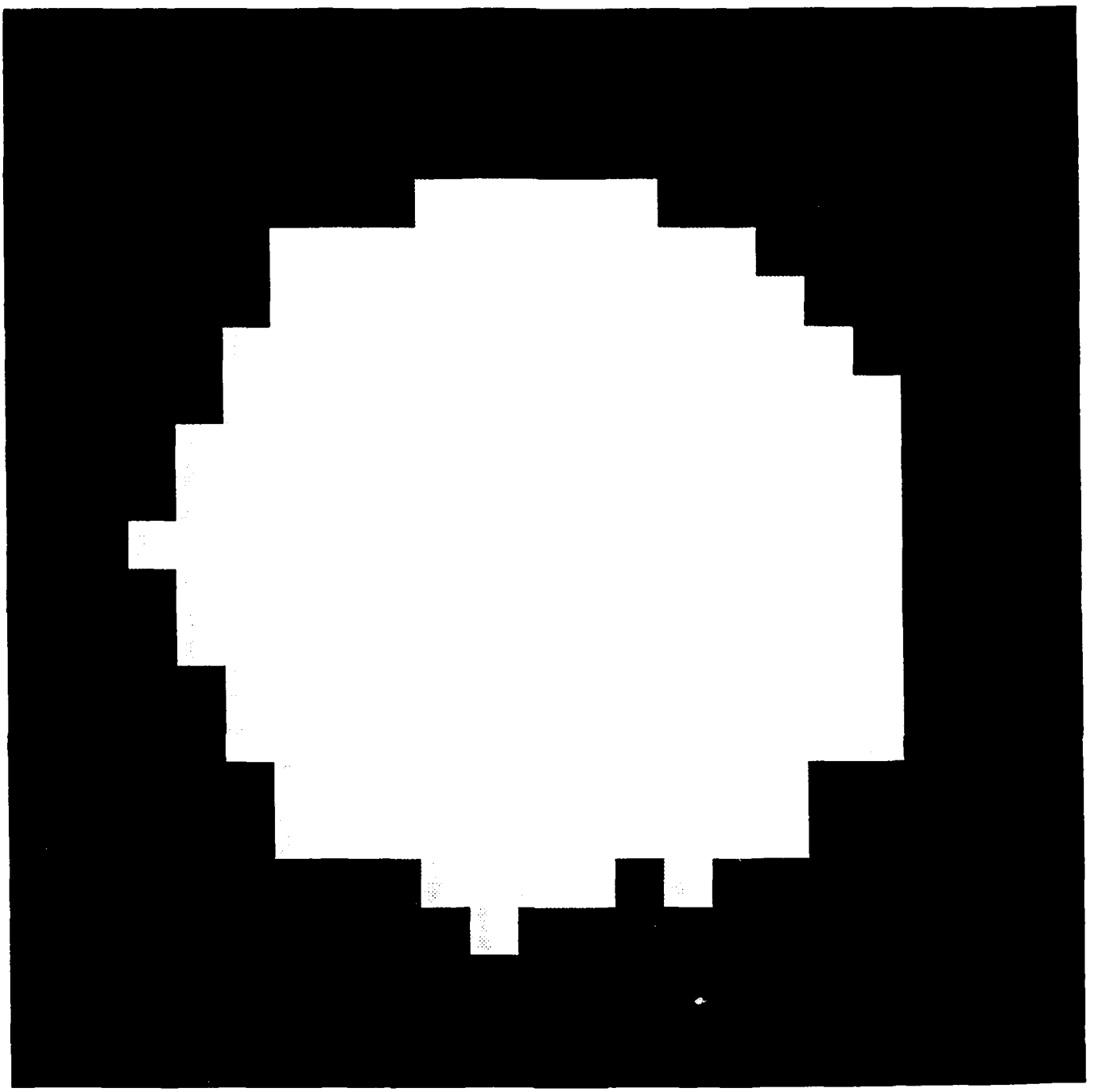
a) 4-sided polygon at SNR = 1.25.



b) 5-sided polygon at SNR = 2.5.



c) 6-sided polygon at SNR = 4.0.



d) 7-sided polygon at SNR = 4.5.

Figure 11. The Segmented Images Used in the Experiment to Determine the Effects of Noise.

6. Summary and Estimate of Technical Feasibility

Our theoretical analysis of the MRP indicates that it estimates the target shape conditions at which a target recognition system will produce a 50% error rate. This can be viewed as an upper bound on the performance of an ATR system. Although performance results are not derived analytically, the simulations analysis do produce empirical relationships between performance and sensor system design parameters. The simulation experiments reported in the previous section validate that the MRP can be used as a reliable measure of the relative performance ability of various pattern recognition systems when compared to human visual pattern recognition.

The MRP relies on a computer simulation of the imaging system. Typically, most imaging system components are described via mathematical/computer models, so this use of a computer simulation to analyze system performance is often preferred to a purely analytical approach. While such a computer simulation requires more extensive computer power than the computation of an analytical formula like the MRT, the MRP may still be considered relatively easy to compute. All software simulations and experiments reported on here were performed on a personal computer.

One significant piece of the software required to automatically perform MRP calculations has been omitted during this phase of the research: the automatic segmentation of noisy images. Although this is an active field of research, a baseline algorithm is required for the computation of the MRP. We have defined such a baseline segmentation algorithm which we have applied manually to segment noisy images. However this algorithm appears to have a straightforward software implementation, therefore it poses no problem to the ultimate use of the MRP.

The MRP analysis could be applied to trade-off studies on essentially any design parameter of the imaging system. The basic performance modeling technique and software which has been developed here can be used to perform sensor system trade-off studies on resolution versus sensitivity, detector shape, and sampling array geometry. In order to perform trade-off studies on sensor parameters which relate to temporal effects such as frame rate (or dwell time), temporal noise (e.g. 1/f noise), and relative motion between the target and sensor only the software simulation of these effects on image pixel intensities needs to be developed.

The computed MRP depends on the use of a new set of synthetic test patterns which we have proposed to replace the bar pattern (which is used to analyze the performance of man-in-the-loop systems). It still must be determined whether predicting system performance based this new test pattern set gives a valid prediction of the system performance in real tactical situations. This may be the biggest challenge to the use of the MRP as a performance measure. However we believe that the new test pattern set has several advantages. It has many of the characteristics of real targets. It is characterized by a single parameter, the number of sides in the polygons. Clutter-like objects are also derived from this test pattern set. And we have defined a logical relationship between the choice of test patterns and the tasks of Detection, Recognition, and Identification.

The MRP could be extended by relating feature computation algorithms to shape distortion. Most feature computations are based on a generic shape model. For example if a target is modeled as a rectangle or ellipse it can be described by the single feature "length to width ratio." Thus given a set of feature values, one could construct a representative shape based on the underlying target model and having the given feature values. This shape can then be compared to the original target to determine distortion. Analyses of this form may lead us to alternative features which better describe target shapes.

Another issue in ATR systems which may be addressed is how to use multi-spectral data. There is already significant research being done on how to use the spectral differences between targets and

clutter for ATR. Another approach would be to use multi-spectral data to improve image shape detection. This may require registered images from different (displaced) sensors or one may consider a single sensor with different detectors for different wavelengths (like the human eye). In the first case one must determine how accurately the images from the different sensors must be registered. In the second case there are issues relating to the relative size, number and location of the detectors for different wavelengths on the focal plane.

7. Appendix

7.1. Derivation of the Polar Coordinate Representation of a General Polygon.

First consider the triangle

$$\rho(\theta,3) = \begin{cases} r/\cos(\theta - 2\pi/6) & \text{for } 0 \le \theta \le 2\pi/3 \\ r/\cos(\theta - 2\pi/6 - 2\pi/3) & \text{for } 2\pi/3 \le \theta \le 4\pi/3 \\ r/\cos(\theta - 2\pi/6 - 4\pi/3) & \text{for } 4\pi/3 \le \theta \le 6\pi/3 \end{cases}$$

r determines the minimum radius of the triangle. The first equation can be obtained from the rectangular coordinates for a vertical line which is a distance r from the origin at its closest point:

$$x = r$$

In polar coordinates this becomes

$$\cos(\theta) = r, \qquad \text{or } \rho = r / \cos(\theta)$$

Each side of the triangle subtends an arc of $2\pi/3$ radians. The arc subtended by this first side is centered on $\theta = 0$. The other sides can be obtained from the first by rotation through $2\pi/3$ and $4\pi/3$ radians to give sides 2 and 3 respectively. With this in mind then, the formula for the regular polygon with n sides can be obtained in polar coordinates.

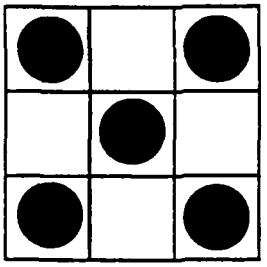
$$\rho(\theta,n) = r/\cos(\theta - 2\pi/2n - k\,2\pi/n)$$

$$\text{for } k\,2\pi/n \le \theta \le (k+1)\,2\pi/n, \quad k=0,1,\ldots,n-1$$

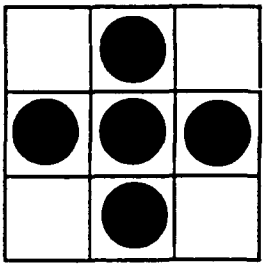
r determines the minimum radius of the polygon, n is the number of sides in the polygon, and k is an index for which side is being drawn, 0 corresponding to the first side, n-1 corresponding to the last side.

7.2. Four-Connected Regions Versus 8-Connected Regions

Two pixels in an 8-connected region are considered adjacent if they touch at either a pixel edge or a pixel corner. The five dark pixels in Figure A-1a are part of a single 8-connected region. Any two pixels in an 4-connected region are considered adjacent only if they touch at a pixel edge. Therefore the five dark pixels in Figure A-1a form five different regions if 4-connectedness is assumed. The five pixels in Figure 9b are part of one region in either 4-connected or 8-connected space. Note that if the target region is defined by 8-connectedness, then the background must be considered to be 4-connected, and, conversely, 4-connected targets can only live in 8-connected backgrounds.



a) The dark pixels are part of a single 8-connected region or form 5 different 4-connected regions.



b) The dark pixels are part of a single 8-connected region or 4-connected region.

Figure A-1. Comparing 4-connected and 8-connected regions

7.3. Software Listings

A FORTRAN listing is attached. The first subroutine, called FUserProcedure, is a control subroutine which is called from *IPLab* with an index to tell it which of the following subroutines the user has selected from the *IPLab* menus. A summary of these subroutines follows:

ReadInputs:	Reads input parameters into local commons
CreateData:	Fills an 512x512 integer image array with a test pattern
OpticalMTF:	Multiplies the image FFT data passed to it by the MTF of the optics. The 2-D FFT is accomplished in *IPLab* before calling this routine.
DetectorMTF:	Multiplies the image FFT data passed to it by the MTF of the optics
Sampler:	Samples the image data on a rectangular or hexagonal lattice
Segmenter:	Thresholds the image data
GetPolarShape:	Computes polar coord. representation of the segmented image
PlotPolarShape:	Places the polar coord. shape data into a vector which can be plotted from within *IPLab*.

The computation of $\mathcal{M}(n)$ is accomplished inside *IPLab* itself.

```
!!M Inlines.t
!!S FUserCodeSegment


  INTEGER*2 FUNCTION FUserProcedure(index,
  1                                 windowKind,
  2                                 dataType,
  3                                 width, height,
  4                                 data0, variables,
  5                                 ROILeft,ROITop,ROIRight,ROIBottom,
  6                                 cTable,
  7                                 imageObject,
  8                                 theMode)

c windowKind
c   0:  image;
c   1:  vector;
c
c dataType
c   0: BYTE image or vector;
c   1: SHORT INTEGER image or vector;
c   2: LONGINT image or vector;
c   3: FLOATING POINT image or vector;
c theMode
c   0: this routine is being called interactively
c   1: this routine is being called from a script

c include 'Types.f'

  STRUCTURE  /Point/
    UNION
     MAP
      INTEGER*2 v
      INTEGER*2 h
     END MAP
     MAP
      INTEGER*2 vh(0:1)
     END MAP
    END UNION
  END STRUCTURE

  STRUCTURE /Rect/
    UNION
      MAP
        INTEGER*2 top, left, bottom, right
      END MAP
      MAP
        RECORD /Point/ topLeft,botRight
      END MAP
    END UNION
  END STRUCTURE

  STRUCTURE /Region/
    INTEGER*2   rgnSize
    record /Rect/ rgnBBox
  END STRUCTURE

  STRUCTURE /RgnPtr/
    POINTER /Region/ RgnP
  END STRUCTURE

  STRUCTURE /RgnHandle/
    POINTER /RgnPtr/ RgnH
  END STRUCTURE

  IMPLICIT    NONE
  INTEGER*2   index,windowKind,dataType,theMode
  INTEGER*4   width,height
  INTEGER*2   data0(width,height)
  INTEGER*4   variables(256)
  INTEGER*4   ROILeft,ROITop,ROIRight,ROIBottom
  INTEGER*2   cTable(4,256)
  RECORD /RgnHandle/ imageObject

  INCLUDE     'Commons.f'

  INTEGER*2   ReadInputs,CreateData,OpticalMTF,DetectorMTF
  INTEGER*2   Sampler,Segmenter,GetPolarShape,PlotPolarShape

    goto(100,200,300,400,500,600,700,800) index
      FUserProcedure = 1
      return

100 continue
      FUserProcedure = ReadInputs(windowKind,dataType,width,height,data0,variables)
    return
200 continue
      FUserProcedure = CreateData(windowKind,dataType,width,height,data0,variables)
    return
300 continue
      FUserProcedure = OpticalMTF(windowKind,dataType,width,height,data0,variables)
    return
```

```
400 continue
      FUserProcedure = DetectorMTF(windowKind,dataType,width,height,data0,variables)
    return
500 continue
      FUserProcedure = Sampler(windowKind,dataType,width,height,data0,variables)
    return
600 continue
      FUserProcedure = Segmenter(windowKind,dataType,width,height,data0,variables)
    return
700 continue
      FUserProcedure = GetPolarShape(windowKind,dataType,width,height,data0,variables)
    return
800 continue
      FUserProcedure = PlotPolarShape(windowKind,dataType,width,height,data0,variables)
    return
900 continue
      FUserProcedure = 1
      call Exit
    return
  end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  integer*2 function ReadInputs(windowKind,dataType,width,height,data0,variables)

  IMPLICIT    NONE
  INTEGER*2   windowKind,dataType
  INTEGER*4   width,height
  REAL*4      data0(width,height)
  INTEGER*4   variables(256)

  REAL*4      IFOVdetx,IFOVdety

  INCLUDE 'Commons.f'

      nPoints    = 1024                          !Number of points around a circluar target
      radius     = 180                           !Radius of standard circular target
      nSides     = variables(0 + 1)              !Number of sides in the polygonal target
      tDiameter  = variables(1 + 1)              !Diameter of circular target (meters)
      tRange     = variables(2 + 1)              !Range to target (meters)
      orient     = abs(variables(3 + 1))         !Rotation angle for target or clutter
      TargOrClut= variables(4 + 1)               ! = 0 to generate a target shape
                                                 ! = 1 to generate clutter shape
      amplitude = variables(5 + 1)/100.0         !Amplitude of random variable added to
                                                 !polygonal shape to make clutter
      bandwidth = variables(6 + 1)/100.0         !Bandwidth of random variable added to
                                                 !polygonal shape to make clutter
      opticalF0 = variables(7 + 1)               !Optical cutoff freq f0 =
                                                 !optical diameter/wavelength (= 10000).
      IFOVdetx  = variables(8 + 1)/100.0         !detector IFOV in x-direction (in mr)
      IFOVdety  = variables(9 + 1)/100.0         !detector IFOV in y-direction (in mr)
      rectOrHex = variables(10 + 1)              ! = 0 for rectangular sampling
                                                 ! = 1 for hexagonal sampling
      SNR       = variables(11 + 1)/100.0        !RMS signal to noise ratio.Must be > 0.
      seed      = variables(12 + 1)              !For random number generator

      radPerPixel = (tDiameter/(2.0*radius))/tRange
      dxSample    = (IFOVdetx*1.0e-3)/radPerPixel
      if(2*(dxSample/2) .eq. dxSample) dxSample = dxSample + 1    !make sure its odd
      dySample    = (IFOVdety*1.0e-3)/radPerPixel
      if(2*(dySample/2) .eq. dySample) dySample = dySample + 1    !make sure its odd

      targValue   = 200
      ReadInputs  = 0
  return
  end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  integer*2 function CreateData(windowKind,dataType,width,height,data0,variables)

c draw the data or clutter as an image in rectangular coordinates.

  IMPLICIT    NONE

  INTEGER*2   forcedDimension
  PARAMETER   (forcedDimension = 512)

  INTEGER*2   windowKind,dataType
  INTEGER*4   width,height
  INTEGER*2   data0(width,height)
  INTEGER*4   variables(256)

  INTEGER*4   loopx,loopy,xIndex,yIndex,index,xValue,yValue,x2,y2,fd2
  INTEGER*2   angleIndex

  REAL*4      rho,theta
  REAL*4      maxRho,minRho
  REAL*4      distance,angle
  REAL*4      pi,twoPI
  REAL*4      Distortion,thetaPrime
  REAL*4      ranU

  INCLUDE 'Commons.f'
```

```
      if((dataType .ne. 1) .or. (windowKind.ne.0)) then      !Only draw into integer image Window
        call SysBeep(5)
        CreateData = 1
        return
      end if
      if((width.ne.forcedDimension) .or. (height.ne.forcedDimension)) then
        call SysBeep(5)
        CreateData = 1
        return
      end if

      pi      = 3.14159265
      twoPI   = 6.2831853
      fd2     =forcedDimension/2
                                                              !First make the snape data
      Distortion  = 0
      maxRho = 0
      minRho = 1.0e32

      do 25 loopx = 1,nPoints
        index = mod(loopx + orient - 1,nPoints) + 1

        if(nSides .eq. 0) then
          shapeData(index)  = radius
        else if(nSides .eq. 3) then
          thetaPrime  = mod(loopx,nPoints/3) * twoPI/nPoints - twoPI/(2*3)
          shapeData(index)  = 2.0*radius/( (1.0 + 1.0/cos(twoPI/(2*3)))*cos(thetaPrime) )
        else if(nSides .eq. 4) then
          thetaPrime  = mod(loopx,nPoints/4) * twoPI/nPoints - twoPI/(2*4)
          shapeData(index)    = 2.0*radius/( (1.0 + 1.0/cos(twoPI/(2*4)))*cos(thetaPrime) )
        else if(nSides .eq. 5) then
          thetaPrime  = mod(loopx,nPoints/5) * twoPI/nPoints - twoPI/(2*5)
          shapeData(index)  = 2.0*radius/( (1.0 + 1.0/cos(twoPI/(2*5)))*cos(thetaPrime) )
        else if(nSides .eq. 6) then
          thetaPrime  = mod(loopx,nPoints/6) * twoPI/nPoints - twoPI/(2*6)
          shapeData(index)  = 2.0*radius/( (1.0 + 1.0/cos(twoPI/(2*6)))*cos(thetaPrime) )
        else
          thetaPrime  = mod(loopx,nPoints/nSides) * twoPI/nPoints - twoPI/(2*nSides)
          shapeData(index)  = 2.0*radius/( (1.0 + 1.0/cos(twoPI/(2*nSides)))*cos(thetaPrime) )
        end if

        if(TargOrClut .eq. 1) then
          Distortion  = (1-bandwidth)*Distortion + bandwidth*(ranU(seed) - .5)
          shapeData(index)  = shapeData(index) + radius*amplitude*Distortion
        end if
        maxRho = max(maxRho,shapeData(index))
        minRho = min(minRho,shapeData(index))
25    continue

                                          !fill space inside rho(theta) with targValue
        do 110 loopx = 1,width
          xIndex = loopx
          xValue = xIndex - fd2
          x2 = xValue**2
          do 110 loopy = 1,height
            yIndex = 1 + height - loopy
            yValue = loopy - fd2
            y2 = yValue**2
            distance = sqrt(float(x2 + y2))
            if(distance .gt. maxRho) then
              data0(xIndex,yIndex) = 0
            else
              if(distance .lt. minRho) then
                data0(xIndex,yIndex) = targValue
              else
                angle = atan2(float(yValue),float(xValue))
                angle = nPoints*(angle + pi)/twoPI
                angleIndex = min(nPoints,max(1,nint(angle)))
                if( distance .le. shapeData(angleIndex) ) then
                  data0(xIndex,yIndex) = targValue
                else
                  data0(xIndex,yIndex) = 0
                end if
              end if
            end if
110     continue
      CreateData = 0
    return
    end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  integer*2 function OpticalMTF(windowKind,dataType,width,height,data0,variables)

c Multiply the Magnitude FFT passed in data0 by the MFT of the optics
c data0 must be REAL*4 data and the FFT is assumed to have DC centered at (1+width/2,1+width/2)

  IMPLICIT    NONE

  INTEGER*2   windowKind,dataType
  INTEGER*4   width,height
  REAL*4      data0(width,height)
```

```
  INTEGER*4   variables(256)

  INTEGER*2   loopx,loopy,centerIndex
  REAL*4      fx,fy,frequency
  REAL*4      fmax,opticFactor,Hopt,omega,twoOverPi

  INCLUDE 'Commons.f'

      if((dataType .ne. 3) .or. (windowKind.ne.0)) then       !Only REAL*4 image data
        call SysBeep(5)
        OpticalMTF = 1
        return
      end if
                                              !Multiply by optical and detector MFTs
      centerIndex     = width/2 + 1
      twoOverPi       = 2.0/PI
      fmax            = 1.0/(2.0*radPerPixel)
      opticFactor     = fmax/(float(width/2) * opticalF0)
      do 50 loopy = 1,height
        do 50 loopx = 1,width
          fx    = loopx - centerIndex
          frequency = sqrt(fx*fx + fy*fy)
          omega = opticFactor*frequency
          if(omega.lt.1.0) then                          !optical MTF
            Hopt  = twoOverPi * (acos(omega) - omega*sqrt(1 - omega*omega))
          else
            Hopt  = 0
          end if
          data0(loopx,loopy) = Hopt * data0(loopx,loopy)
50    continue

      OpticalMTF = 0
    return
    end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  integer*2 function DetectorMTF(windowKind,dataType,width,height,data0,variables)

c Multiply the Magnitude FFT passed in data0 by the MFT of the detector
c data0 must be REAL*4 data and the FFT is assumed to have DC centered at (1+width/2,1+width/2)

  IMPLICIT    NONE

  INTEGER*2   windowKind,dataType
  INTEGER*4   width,height
  REAL*4      data0(width,height)
  INTEGER*4   variables(256)

  INTEGER*2   loopx,loopy,centerIndex
  REAL*4      fx,fy,frequency
  REAL*4      detFactor,omegax,omegay,Hdetx,Hdety
  REAL*4      temp

  INCLUDE 'Commons.f'

      if((dataType .ne. 3) .or. (windowKind.ne.0)) then       !Only REAL*4 image data
        call SysBeep(5)
        DetectorMTF = 1
        return
      end if
                                              !Multiply by detector MFT
      centerIndex     = width/2 + 1
      detFactor       = PI/float(width)
      do 50 loopy = 1,height
                                                      !y-dimension detector MFT
        fy = loopy - centerIndex
        omegay  = detFactor*fy
        temp    = sin(omegay)
        if(temp.ne.0) then
          Hdety = sin(dySample*omegay)/(dySample*temp)
        else
          Hdety = 1
        end if
        do 50 loopx = 1,widtn
                                                      !x-dimension detector MFT
          omegax  = detFactor*fx
          temp    = sin(omegax)
          if(temp.ne.0) then
            Hdetx = sin(dxSample*omegax)/(dxSample*temp)
          else
            Hdetx = 1
          end if
          data0(loopx,loopy) = Hdety * Hdetx * data0(loopx,loopy)
50    continue

      DetectorMTF = 0
    return
    end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  integer*2 function Sampler(windowKind,dataType,width,height,data0,variables)
```

```
  IMPLICIT    NONE

  INTEGER*2   windowKind,dataType
  INTEGER*4   width,height
  INTEGER*2   data0(width,height)
  INTEGER*4   variables(256)

  INTEGER*2   loopx,loopy,xIndex,yIndex,xsIndex,etaIndex,xTest,yTest
  REAL*4      noiseFactor
  REAL*4      RanG

  INCLUDE 'Commons.f'

  INTEGER*2   SampleIndexRect

      if((dataType .ne. 1) .or. (windowKind.ne.0)) then       !Only sample integer image data
        call SysBeep(5)
        Sampler = 1
        return
      end if
                                                              !sample image and add noise
      noiseFactor = float(targValue)/SNR
                                                                !Rectangular array
      if(rectOrHex.eq.0) then
        do 50 loopx = 1,width
          xIndex = SampleIndexRect(loopx,dxSample)
          do 50 loopy = 1,height
            yIndex = SampleIndexRect(loopy,dySample)
            if((xIndex.ne.loopx) .or. (yIndex.ne.loopy)) then
              data0(loopx,loopy) = 0
            else
              data0(loopx,loopy) = data0(loopx,loopy) + nint(noiseFactor*rang(seed))
            end if
50      continue
                                                              !Reconstruction to original size
        do 75 loopx = 1,width
          xIndex = SampleIndexRect(loopx,dxSample)
          do 75 loopy = 1,height
            yIndex = SampleIndexRect(loopy,dySample)
            data0(loopx,loopy) = data0(xIndex,yIndex)
75      continue

      else                                                    !Hexagonal array
        do 100 loopx = 1,width
          do 100 loopy = 1,height
            call SampleIndexHex(loopx,loopy,xIndex,yIndex,dxSample,dySample)
            if((xIndex.ne.loopx) .or. (yIndex.ne.loopy)) then
              data0(loopx,loopy) = 0
            else
              data0(loopx,loopy) = data0(loopx,loopy) + nint(noiseFactor*rang(seed))
            end if
100     continue
                                                              !Reconstruction to original size
        do 175 loopx = 1,width
          do 175 loopy = 1,height
            call SampleIndexHex(loopx,loopy,xIndex,yIndex,dxSample,dySample)
            data0(loopx,loopy) = data0(xIndex,yIndex)
175     continue
      end if

      Sampler = 0
    return
    end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  subroutine SampleIndexHex(xCoord,yCoord,xIndex,yIndex,dxSample,dySample)
    IMPLICIT    NONE
    integer*2   xCoord,yCoord,xIndex,yIndex,dxSample,dySample
    integer*2   ds2

      yIndex  = inint(float(yCoord - 1)/float(dySample))
      if(mod(yIndex,2).eq.0) then
        ds2 = dxSample/2 + 1
        xIndex  = dxSample * inint(float(xCoord - ds2)/float(dxSample)) + ds2
      else
        xIndex  = dxSample * inint(float(xCoord - 1)/float(dxSample)) + 1
      end if
      yIndex  = dySample * yIndex + 1
    return
    end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  integer*2 function SampleIndexRect(coord,dSample)
    IMPLICIT    NONE
    integer*2   coord,dSample

    SampleIndexRect = dSample * nint(float(coord - 1)/float(dSample)) + 1
    return
    end


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
  integer*2 function Segmenter(windowKind,dataType,width,height,data0,variables)
!!SetC usingincludes = false
    include 'Types.f'
    include 'QuickDraw.f'

c Segment image data.

  IMPLICIT    NONE

  INTEGER*2   windowKind,dataType
  INTEGER*4   width,height
  INTEGER*2   data0(width,height)
  INTEGER*4   variables(256)

  INTEGER*2   loopx,loopy,threshold,gotAnObject
  RECORD /RgnHandle/ wholeImage,tempObject,bigObject
  INTEGER*2   left,top,right,bottom       !local variables for the region bounding rectangle
  RECORD /Point/  myPoint

  INCLUDE 'Commons.f'

                                                          !First threshold
    threshold = targValue/2
    do 100 loopy = 1,height
      do 100 loopx = 1,width
        if(data0(loopx,loopy) .gt.threshold) then
          data0(loopx,loopy) = targValue
        else
          data0(loopx,loopy) = 0
        end if
100 continue


1000    Segmenter = 0
    return
    end

ccccccccccc:ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  integer*2 function GetPolarShape(windowKind,dataType,width,height,data0,variables)

c Convert image shape to vector using polar coordinates.

  IMPLICIT    NONE

  INTEGER*2   forcedDimension,fd2
  PARAMETER   (forcedDimension = 512)

  INTEGER*2   windowKind,dataType
  INTEGER*4   width,height
  INTEGER*2   data0(width,height)
  INTEGER*4   variables(256)

  INTEGER*2   loopTheta,loopRho,maxLoopRho,xIndex,yIndex

  REAL*4      theta,deltaTheta,cosTheta,sinTheta
  REAL*4      rho,rhoMax
  REAL*4      x,y
  REAL*4      pi,twoPI

  INCLUDE 'Commons.f'

      if((dataType.ne.1) .or. (windowKind.ne.0)) then       !Only into integer image Window
        call SysBeep(5)
        GetPolarShape = 1
        return
      end if
      if((width.ne.forcedDimension) .or. (height.ne.forcedDimension)) then
        call SysBeep(5)
        GetPolarShape = 1
        return
      end if

      pi      = 3.14159265
      twoPI   = 6.2831853
      deltaTheta  = twoPi/float(nPoints)
      fd2     = forcedDimension/2
      rhoMax      = sqrt(2.0)*fd2
                                                          !loop on theta index
      do 110 loopTheta = 1,nPoints
        theta     = (loopTheta - 1)*deltaTheta
        cosTheta  = cos(theta)
        sinTheta  = sin(theta)
                                                          !loop that direction to find edge
        do 50 loopRho = 1,rhoMax
          x       = cosTheta*(loopRho - 1)
          y       = sinTheta*(loopRho - 1)
          xIndex  = nint(x) + fd2
          yIndex  = fd2 - nint(y)
          if((xIndex.le.1) .or. (xIndex.ge.forcedDimension)) goto 75
          if((xIndex.le.1) .or. (xIndex.ge.forcedDimension)) goto 75
          if(data0(xIndex,yIndex).lt.targValue/2) goto 75
50      continue
```

```
75      snapeData(loopTheta) = sqrt(x*x + y*y)
110   continue
      GetPolarShape = 0
      return
      end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
  integer*2 function PlotPolarShape(windowKind,dataType,width,height,data0,variables)

c Transfer data to the vector in data0.

  IMPLICIT    NONE

  INTEGER*2   windowKind,dataType
  INTEGER*4   width,height
  REAL*4      data0(width,height)
  INTEGER*4   variables(256)

  INTEGER*2   loop

  INCLUDE 'Commons.f'

      if((dataType.ne.3) .or. (windowKind.ne.1)) then       !Only into floating vector Window
        call SysBeep(5)
        PlotPolarShape = 1
        return
      end if
      if((width.lt.nPoints)) then
        call SysBeep(5)
        PlotPolarShape = 1
        return
      end if

      do 110 loop = 1,nPoints
        data0(loop,1) = snapeData(loop)
110   continue
      PlotPolarShape = 0
      return
      end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      real*4 function RanU(seed)
c
c returns a uniformly distributed random variable between 0.0 and 1.0
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      real*8    seed,t,a,b
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c These are the parameters used by VAX11/780 VMS t=2**32 so the period of
c this sequence should be 2**32.
c An alternative value for a is 1664525 as given in Knuth. It scores better
c on the spectral test than a=69069.
      t=4294967296.0d0
      a=69069.0d0
      seed=dmod(a*seed + 1.0d0,t)
      ranu=seed/t
      return
      end


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      real*4 function RanG(seed)
c
c compute a gaussian distributed random variable (NOTE:  seed is do  le
c precision)
c by m.mort
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      real*8    seed
      real*4    u1,u2
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      u1 = RanU(seed)
      u2 = RanU(seed)
      if(u1.ne. 0.0) then
       RanG = sqrt(-2.0*alog(u1))*cos(6.283185307*u2)
      else
       RanG = sqrt(73.0)*cos(6.283185307*u2)
      end if
      return
      end
```